SEDIANE CARMEM LUNARDI HERNANDES

Um Modelo de Comunicação de Eventos para *Middlewares Publish-Subscribe* por Aptidão de Assinante em Cidades Inteligentes

Curitiba

2019

SEDIANE CARMEM LUNARDI HERNANDES

Um Modelo de Comunicação de Eventos para *Middlewares Publish-Subscribe* por Aptidão de Assinante em Cidades Inteligentes

Tese apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Doutor em Informática.

Pontifícia Universidade Católica do Paraná

Programa de Pós-Graduação em Informática

Orientador: Marcelo Eduardo Pellenz

Coorientador: Alcides Calsavara

Curitiba

2019

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Luci Eduarda Wielganczuk – CRB 9/1118

H557m
2019

Hernandes, Sediane Carmem Lunardi
Um modelo de comunicação de eventos para *middlewares publish-subscribe* por aptidão de assinante em cidades inteligentes / Sediane Carmem Lunardi Hernandes ; orientador: Marcelo Eduardo Pellenz ; coorientador: Alcides Calsavara. – 2019.
145 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2019
Bibliografia: f. 130-136

1. Middleware. 2. Modelo de comunicação baseado em eventos. 3. Planejamento urbano – Inovações tecnológicas. I. Pellenz, Marcelo Eduardo. II. Calsavara. Alcides. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

CDD 20. ed. – 004.65

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

DECLARAÇÃO

Declaro para os devidos fins que a aluna **SEDIANE CARMEM LUNARDI HERNANDES**, defendeu sua tese de doutorado intitulada **"Um Modelo de Comunicação de Eventos para Middlewares *Publish-Subscribe* por Aptidão de Assinante em Cidades Inteligentes"**, na área de concentração Ciência da Computação, no dia 06 de dezembro de 2019, no qual foi aprovada.

Declaro ainda que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade, firmo a presente declaração.

Curitiba, 10 de fevereiro de 2020.

Programa de Pós-Graduação em Informática
PPGIa
PUCPR

Prof. Dr. Emerson Cabrera Paraiso
Coordenador do Programa de Pós-Graduação em Informática
Pontifícia Universidade Católica do Paraná

Rua Imaculada Conceição, 1155 Prado Velho CEP 80215 901 Curitiba Paraná Brasil
Tel.: (41) 3271 1669 Fax.: (41) 3271 2121 www.ppgia.pucpr.br

Dedico este trabalho aos meus filhos Felipe e Rafael, ao meu esposo Fábio, aos meus pais e avós.

Agradecimentos

Em primeiro lugar, agradeço a Deus pelo dom da vida, por seu amor e misericórdia. Ao meu esposo Fábio, obrigada por ter sido minha força durante esses anos de doutorado, e aos meus filhos Felipe e Rafael por me ensinarem que o mais importante é invisível aos olhos. Aos meus pais Alvino e Clair Inês, ao meu irmão Fabiano e a minha avó, Maria Corbelini Scartezini, meu muito obrigada por todas as orações, pelo apoio e incentivo. A todos os meus familiares, gratidão!

De forma sincera, agradeço a todos os funcionários e professores da Pontifícia Universidade Católica do Paraná, em especial aos professores Marcelo Eduardo Pellenz e Alcides Calsavara. Ao professor Alcides meu profundo obrigado pela atenção e dedicação no acompanhamento do trabalho. Ao meu orientador, professor Marcelo, minha gratidão pela atenção, dedicação, ajuda e empenho para que o trabalho tivesse bons resultados e produzisse publicações de qualidade para o Programa de Pós-Graduação. Aos amigos e colegas do PPGIa, obrigada!

Agradeço aos servidores técnico administrativos e professores da Universidade Tecnológica Federal do Paraná - Câmpus Guarapuava por autorizarem o afastamento. À professora Sílvia do Nascimento Rosa, minha colega de trabalho e amiga, obrigada por me ouvir muitas vezes e pelos bons conselhos. Aos amigos de longa data que fizeram parte da minha vida, gratidão pelos bons momentos, pelos risos e alegrias.

Agradeço a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pela bolsa auxílio mensalidade obtida por meio do Programa de Apoio à Pós-Graduação em Instituições Comunitárias de Ensino Superior (PROSUC) e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela ajuda de custo para participar de evento.

Mais uma etapa chega ao fim e tenho a certeza de que novos desafios estarão me aguardando. Gratidão a todos que de longe ou de perto estiveram comigo nessa jornada que se chama Doutorado!

Povos todos, louvai o SENHOR,
nações todas, dai-lhe glória;
porque forte é seu amor para conosco
e a fidelidade do SENHOR dura para sempre.
(*Salmo 116*)

Resumo

A população mundial está crescendo e com isso novos desafios precisam ser enfrentados em todo o mundo, especialmente em cidades maiores e mais populosas. Os principais problemas estão relacionados ao fornecimento de infraestrutura e serviços públicos. Cidades Inteligentes exploram as Tecnologias de Informação e Comunicação - TICs, com o objetivo de fazer melhor uso dos recursos públicos para aprimorar a qualidade dos serviços oferecidos aos cidadãos melhorarnado a qualidade de vida dos mesmos. Novas estratégias para gerenciar as cidades com a ajuda das TICs surgem, dentre elas a Internet das Coisas (*Internet of Things - IoT*) aplicada em um contexto urbano (*urban IoT*). IoT é uma rede de objetos inteligentes interconectados em que os objetos se comunicam entre si, tomam decisões com base em dados recebidos, reagem a eventos, entre outros. Como administrar esses eventos é uma questão a ser gerenciada. Um *middleware* para administrar o grande número de eventos que os objetos inteligentes, presentes no ambiente urbano, geram espontaneamente pode ser utilizado. Intuitivamente, um *middleware* fornece um mecanismo de seleção e outro de entrega de eventos. No entanto, durante a entrega de um evento pode ser necessário que somente determinado número de assinantes, os melhores dentro de um contexto específico (p. e., uma ambulância com UTI móvel a ser utilizada em uma situação de emergência particular), recebam aquele evento e o trate da maneira mais rápida possível. Um assinante pode apresentar melhores condições para receber o evento e tratá-lo do que os outros. Em um *middleware publish-subscribe*, essa possibilidade de notificação de eventos a um subconjunto de assinantes dentre todos os que se inscreveram para o evento não é possível. Desta forma, o objetivo deste trabalho é a especificação de um novo modelo de comunicação de eventos para *middlewares publish-subscribe* aplicado a serviços em Cidades Inteligentes. O novo modelo chamado *Event to Most Suitable Subscribers (EMSS)* permite o envio de eventos por aptidão de assinante, ou seja, o envio de eventos a um subconjunto dos melhores assinantes (i. e., os mais adequados ou mais aptos) do serviço necessário para tratar o evento (p. e., o serviço policial). EMSS assume como base para este envio, o cálculo de um valor de aptidão (*fitness*) para cada assinante de um serviço que permite identificar os melhores segundo critérios específicos do serviço. O estudo foi planejado e conduzido para testar a seguinte hipótese de pesquisa: Se for utilizado o modelo de comunicação de eventos proposto, utilizando uma infraestrutura fixa e assinantes móveis geograficamente distribuídos em uma cidade, então haverá ganho de tempo no atendimento aos eventos em relação a uma escolha entre uma porcentagem dos melhores assinantes disponíveis. Para alcançar o objetivo e confirmar ou refutar a hipótese de pesquisa, a simulação foi utilizada devido às limitações na utilização de outros métodos controlados para alcançar o objetivo proposto. Um *middleware publish-subscribe* distribuído para validar o modelo de eventos proposto foi implementado e recebeu o mesmo nome do modelo de comunicação. Simulações dos serviços policial, bombeiro e transporte

de passageiros, utilizando o *middleware* implementado em um ambiente simulado de Cidades Inteligentes, foram realizadas considerando três abordagens diferentes: Proativa mais apropriada (PMS), Reativa mais aproviada (RMS) e Reativa aleatória (RR). Na primeira abordagem, o valor de aptidão de cada assinante é atualizado proativamente, ou seja, antes do evento ocorrer. Na abordagem RMS e RR, o valor de *fitness* é calculado em resposta a ocorrência de um evento. As simulações foram realizadas e elaboradas análises dos dados coletados. As análises comprovaram a hipótese de pesquisa levantada em que os assinantes mais adequados atendem os eventos em um espaço de tempo menor.

Palavras-chave: Cidades Inteligentes, *Middleware publish-subscribe*, Modelo de comunicação baseado em eventos.

Abstract

The world's population is growing and as a result, new challenges need to be faced around the world, especially in larger and more populous cities. The main problems are related to the provision of infrastructure and public services. Smart Cities exploit Information and Communication Technologies (ICTs) with the aim of making better use of public resources to improve the quality of services offered to citizens and improve their quality of life. New strategies for managing cities with the help of ICTs emerge, among them the Internet of Things (IoT) applied in an urban context (IoT). IoT is a network of interconnected smart objects in which objects communicate with each other, make decisions based on received data, react to events, among others. How to handle these events is a question to be managed. A middleware to manage the large number of events that the smart objects, present in the urban environment, generate spontaneously can be used. Intuitively, a middleware provides mechanisms for selection and event delivery. However, during the delivery of an event, it may be necessary that only a certain number of the most suitable subscribers, the best within a specific context (e. g., a mobile ICU ambulance to be used in a particular emergency), receive that event as quickly as possible. A subscriber may present better conditions to receive the event and treat it than others. In a middleware, this possibility of event notification to a subset of subscribers among all those who have subscribed for the event is not possible. In this way, the objective of this work was the specification of a new event model (i.e., communication model) for publish-subscribe middlewares applied to services in Smart Cities. The new model called *Event to Most Suitable Subscribers (EMSS)* allows the sending of events by subscriber suitability, that is, the sending of events to a subset of the most appropriate subscribers (i. e., the most suitable or fit) of the service required to handle the event (e. g., the police service). EMSS assumes as the basis for this send, the calculation of a fitness value (*fitness*) for each subscriber of a service that allows to identify the best according to specific criteria of the service. The study was planned and conducted to test the following research hypothesis: If the proposed event communication model is used, using a fixed infrastructure and geographically distributed mobile subscribers in a city, then time will be saved in attending events compared to choosing from a percentage of the best available subscribers. In order to reach the objective and confirm or refute the research hypothesis, the simulation was used due to the limitations in the use of other controlled methods to reach the proposed objective. A distributed publish-subscribe middleware to validate the proposed event model has been deployed and received the same name as the communication model. Simulations of the police, firefighter, and passenger transport services, using the middleware implemented in a simulated environment of Smart Cities, were performed considering three different approaches: Proactive Most Suitable (PMS), Reactive Most Suitable (RMS) and Reactive Random (RR). In the first approach, the *fitness* value of each subscriber is proactively

updated, i. e., before the event occurs. In the RMS and RR approach, the *fitness* value is calculated in response to an event occurring. The simulations were performed and analyzes of the collected data were elaborated. The analysis has proven the research hypothesis raised in which the most suitable subscribers attend the events in a shorter period of time.

Keywords: Smart Cities, Middleware publish-subscribe, Event-based communication model.

Lista de ilustrações

Figura 1 – (a) Padrão de comunicação *Device-to-Gateway* (b) Padrão de comunicação *Back-end Data Sharing*. 30
Figura 2 – Arquitetura em camadas de uma Cidade Inteligente Genérica. 32
Figura 3 – Arquitetura de *Fog Computing*. 34
Figura 4 – Infraestrutura do *testbed* SmartSantander. 42
Figura 5 – Plataforma CiDAP construída sobre o *testbed* SmartSantander. 43
Figura 6 – Demonstração de nós sensores implantados em uma avenida com tráfego intenso. 44
Figura 7 – Localidades em que o programa Wifi LivreSP se faz presente: Norte (Verde), Sul (Roxo), Leste (Laranja), Oeste (Azul), Centro (Vermelho). 46
Figura 8 – Modelo de projeto para um *middleware* baseado em eventos em IoT. . 48
Figura 9 – Visão geral de um *middleware* baseado em eventos. 49
Figura 10 – Exemplo de um filtro de evento. 50
Figura 11 – Método básico de interação em um serviço de eventos. 52
Figura 12 – Arquitetura do broker em um *middleware publish-subscribe*. 56
Figura 13 – Serviço de notificação de eventos distribuído. 57
Figura 14 – Topologias: a) Hierárquica; b) *Peer-to-peer* acíclica; c) *Peer-to-peer* geral. 59
Figura 15 – Exemplos de topologias híbrida. 59
Figura 16 – Exemplos de algoritmos de eventos (a) *Event flooding* e (b) *Subscription flooding*. 61
Figura 17 – Exemplos de algoritmos de eventos *Rendezvous* e baseado em filtro. . . 63
Figura 18 – Arquitetura do *middleware* EMSS. 82
Figura 19 – Redes overlay criadas no *broker* b_1. 83
Figura 20 – Diagrama de Estados do *broker* representando os estados quando recebe uma mensagem de subscrição de um assinante. 84
Figura 21 – Diagrama de Estados do *broker* representando os estados quando recebe uma mensagem de *bind* de um assinante. 89
Figura 22 – Diagrama de sequência relacionado a troca das mensagens *subscribe*, *bind* e *unsubscribe*. 90
Figura 23 – Diagrama de Estados referente a evento recebido por *broker* publicador 93
Figura 24 – Diagrama de sequência relacionada a notificação de eventos (abordagem proativa). 94
Figura 25 – Exemplo de Rede Overlay no broker b_2. 95
Figura 26 – Diagrama de sequência relacionada a notificação de eventos (abordagem reativa). 96
Figura 27 – Diagrama de classes do *middleware* EMSS. 97

Figura 28 – Área da cidade simulada. 103
Figura 29 – Comparação do TRS nas abordagens PMS, RMS e RR para o serviço de bombeiro. 111
Figura 30 – Média dos valores de *fitness* para o serviço de bombeiro. 112
Figura 31 – Comparação do TRS nas abordagens PMS, RMS e RR para o serviço policial. 113
Figura 32 – Média dos valores de *fitness* das abordagens PMS, RMS e RR para o serviço policial. 114
Figura 33 – Média do número de unidades móveis ativadas de acordo com o nível de avaliação do motorista para o serviço de transporte. 116
Figura 34 – Comparação das abordagens PMS, RMS e RR em relação ao número de unidades móveis disponíveis para o serviço de transporte. 117
Figura 35 – Comparação do *fitness* para as abordagens PMS, RMS e RR em relação ao número de unidades móveis disponíveis para o serviço de transporte. 118
Figura 36 – **Tempo de Resposta do Serviço** das abordagens PMS, RMS e RR para o serviço de transporte. 119
Figura 37 – Média do *fitness* nas abordagens PMS, RMS e RR para o serviço de transporte. 120
Figura 38 – Número total de mensagens trocadas na abordagem PMS para o serviço policial. 124
Figura 39 – Média do número total de mensagens trocadas nas abordagens RMS e RR para o serviço policial. 124
Figura 40 – Número total de mensagens trocadas na abordagem PMS para o serviço de bombeiro. 125
Figura 41 – Média do número total de mensagens trocadas nas abordagens RMS e RR para o serviço de bombeiro. 125
Figura 42 – Percentual de ocorrência por tipo para o serviço bombeiro. 140
Figura 43 – Percentual de ocorrência por dia da semana para o serviço bombeiro. . 141
Figura 44 – Distribuição de dados para o tempo de deslocamento do serviço bombeiro. 141
Figura 45 – Distribuição de dados para o tempo de atendimento do serviço bombeiro. 143
Figura 46 – Percentual de ocorrência por tipo para o serviço policial. 143
Figura 47 – Distribuição do tempo de atendimento para o serviço policial. 144

Lista de tabelas

Tabela 1 – Principais *Middlewares Publish-Subscribe*. 21
Tabela 2 – Exemplo de *matching*. 51
Tabela 3 – Comparação entre *Middlewares Publish-Subscribe*. 73
Tabela 4 – Denotação das entidades do modelo EMSS. 77
Tabela 5 – Definições do modelo EMSS. 78
Tabela 6 – Parâmetros do modelo de chamados criado para a geração dos históricos. 105
Tabela 7 – Definição do coeficiente de tráfego c_{ij}^{k}. 106
Tabela 8 – Condições de trânsito de uma cidade real. 107
Tabela 9 – Cenários simulados nos serviços de emergência. 109
Tabela 10 – Cenários simulados no serviço de transporte. 109
Tabela 11 – Médias dos valores de *fitness* máximo e mínimo para o serviço de bombeiro. 112
Tabela 12 – Médias das distâncias percorridas para tratar chamadas de emergência no serviço de bombeiro. 113
Tabela 13 – Médias das distâncias percorridas para tratar chamadas de emergência no serviço policial. 115
Tabela 14 – Médias dos *Fitness* máximo e mínimo para o serviço policial. 115
Tabela 15 – Médias das distâncias percorridas para tratar chamadas de passageiros no serviço de transporte. 117
Tabela 16 – Média dos valores mínimo e máximo de *fitness* para o serviço de transporte. 118
Tabela 17 – Média das distâncias percorridas nas abordagens RMS e RR para o serviço de transporte. 120
Tabela 18 – Visão do número total de mensagens trocadas nas diferentes abordagens para uma única simulação. 121
Tabela 19 – Percentual de ocorrências por dia do mês para o serviço bombeiro. . . 139
Tabela 20 – Percentual de ocorrências por hora do dia para o serviço bombeiro. . . 142
Tabela 21 – Percentual de ocorrências por mês do ano para o serviço policial. . . . 144

Sumário

1 INTRODUÇÃO . . . 17
1.1 Objetivo . . . 19
1.2 Hipótese de Pesquisa . . . 19
1.3 Justificativa e Problematização . . . 19
1.4 Contribuição . . . 24
1.5 Publicações . . . 24
1.6 Suporte Financeiro . . . 25
1.7 Divisão do Trabalho . . . 25

2 CIDADES INTELIGENTES . . . 26
2.1 Internet das Coisas como Infraestrutura de Cidades Inteligentes . . . 27
2.1.1 Características de IoT . . . 27
2.1.2 Padrões de Comunicação para Objetos Inteligentes em IoT . . . 29
2.1.2.1 Padrão de Comunicação Device-to-Device . . . 29
2.1.2.2 Padrão de Comunicação Device-to-Cloud . . . 29
2.1.2.3 Padrão de Comunicação Device-to-Gateway . . . 30
2.1.2.4 Padrão de Comunicação Back-End Data Sharing . . . 30
2.2 Arquiteturas para Cidades Inteligentes . . . 31
2.2.1 Fog Computing como parte da Camada de Transmissão . . . 33
2.3 Aplicações de Cidades Inteligentes . . . 35
2.3.1 Serviços de Emergência . . . 39
2.4 Cidades Inteligentes Espalhadas pelo Mundo . . . 41
2.4.1 Santander (Espanha) . . . 41
2.4.2 Padova (Itália) . . . 44
2.4.3 Amsterdam (Holanda) . . . 45
2.4.4 Chicago (Estados Unidos) . . . 45
2.4.5 São Paulo (Brasil) . . . 46

3 SERVIÇO DE NOTIFICAÇÃO DE EVENTOS . . . 47
3.1 Conceito . . . 48
3.1.1 Evento . . . 50
3.1.2 Filtros de Eventos . . . 50
3.1.3 Matching em um Serviço de Eventos Publish-subscribe . . . 51
3.2 Método Básico de Interação Publish-subscribe . . . 51
3.3 Modelos de Subscrição . . . 53
3.3.1 Publish-Subscribe baseado em Tópicos . . . 53

3.3.2 Publish-Subscribe baseado em Conteúdo ou Propriedade . . . 53
3.3.3 Publish-Subscribe baseado em Tipo . . . 54
3.4 Arquitetura de um Serviço de Notificação de Eventos . . . 55
3.5 Topologia de Brokers na Organização Distribuída . . . 58
3.6 Roteamento de Eventos . . . 60
3.6.1 Algoritmos de Roteamento Event flooding e Subscripition flooding . . . 61
3.6.2 Roteamento de Eventos Seletivo . . . 62
3.6.2.1 Roteamento baseado em *Rendezvous* . . . 62
3.6.2.2 Roteamento baseado em Filtro ou Conteúdo . . . 63
3.6.2.3 Roteamento baseado em Gossip . . . 64
3.7 Trabalhos Relacionados a Middlewares baseados em Eventos . . . 65

4 O MODELO DE EVENTOS *EVENT TO MOST SUITABLE SUBSCRIBERS* - EMSS . . . 75
4.1 Descrição Formal do Modelo EMSS . . . 77
4.2 O Middleware EMSS . . . 80
4.2.1 Arquitetura do Serviço de Notificação de Eventos EMSS . . . 81
4.2.2 Redes Overlay . . . 82
4.2.3 Subscrições para Eventos . . . 83
4.2.4 Notificação de Eventos . . . 89
4.2.4.1 Notificação na Abordagem Proativa . . . 89
4.2.4.2 Notificação na Abordagem Reativa . . . 94
4.2.5 Mensagens das Entidades do Middleware EMSS . . . 96

5 AVALIAÇÃO DO MODELO DE EVENTOS EMSS . . . 100
5.1 Método de Pesquisa . . . 100
5.2 Simulador . . . 101
5.2.1 Estudos de Caso Simulados . . . 102
5.2.2 Área da Cidade Simulada . . . 102
5.2.3 Chamados de Eventos . . . 103
5.2.4 Condições de Trânsito . . . 106
5.2.5 Parâmetros do Simulador . . . 106
5.2.6 Métricas de Desempenho . . . 108
5.2.7 Número de Simulações . . . 109
5.3 Resultados . . . 110
5.3.1 Serviço de Bombeiro . . . 110
5.3.1.1 Comparação dos Tempos de Resposta do Serviço . . . 110
5.3.2 Serviço Policial . . . 112
5.3.2.1 Comparação dos Tempos de Resposta do Serviço . . . 113
5.3.3 Serviço de Transporte . . . 115

5.3.3.1 Comparação do Número de Unidades Móveis Ativadas 115
5.3.3.2 Comparação dos Tempos de Resposta do Serviço 116
5.3.4 Comparação do Número de Mensagens 120

6 CONCLUSÕES 128

REFERÊNCIAS 130

ANEXOS 137

ANEXO A – DADOS COLETADOS 138
A.1 Corpo de Bombeiros 138
A.1.1 Descrição dos dados 138
A.2 Policial 142
A.3 Transporte de Passageiros 145

1 Introdução

No Brasil, em meados dos anos 50, a população que vivia em áreas rurais era muito maior do que a urbana (IBGE, 2012). Em 2000, o Censo Demográfico do Brasil apontou que o território brasileiro era composto de 170 milhões de pessoas e no Censo de 2010 de 190 milhões de pessoas. Um aumento de aproximadamente 12% no número de habitantes entre 2000 e 2010. Contudo, o número de pessoas continua a crescer e a população rural está cada vez mais se mudando para as cidades. Para 2030 é estimado que mais de 90% da população brasileira viva nas cidades e que em 2050 toda a América Latina seja 86% urbana (ONU, 2012). No mundo esse crescimento populacional não é diferente, a população que vive nas cidades atualmente passa dos 50% e para 2050 é esperado que esse percentual aumente para 70% (ALKANDARI; ALNASHEET; ALSHAIKHLI, 2012). Com isso, novos desafios precisam ser enfrentados em todo o mundo, especialmente em cidades maiores e mais populosas. Os principais problemas estão relacionados ao fornecimento de infraestrutura e serviços públicos, como gerenciamento de resíduos, escassez de recursos, poluição do ar, preocupações com a saúde, gestão do tráfego urbano, educação, desemprego, entre outros.

Nesse contexto, o conceito de Cidades Inteligentes (*Smart Cities*) consiste em explorar as Tecnologias de Informação e Comunicação – TICs (ICTs – *Information Communication Technologies*) com o objetivo de fazer melhor uso dos recursos públicos para aprimorar a qualidade dos serviços oferecidos aos cidadãos e, como consequência, melhorar a qualidade de vida dos mesmos (CENEDESE et al., 2014). Novas estratégias para gerenciar as cidades com a ajuda das TICs surgem, dentre elas a Internet das Coisas (*Internet of Things - IoT*) ou Internet dos objetos inteligentes. IoT é uma rede de objetos físicos interconectados (objeto inteligentes) incluindo computadores, smartphones, sensores, atuadores, casas, edifícios, estruturas, veículos e sistemas de energia (MOHANTY; CHOPPALI; KOUGIANOS, 2016). Em IoT, objetos inteligentes se comunicam entre si, tomam decisões com base em dados recebidos, reagem a eventos, entre outros. Logo, *urban IoT* trata do uso de objetos inteligentes em um contexto urbano de modo a oferecer serviços que venham a auxiliar a administração pública das cidades, as empresas e os cidadãos (ZANELLA et al., 2014).

IoT aplicada em um contexto urbano (*urban IoT*) responde aos interesses governamentais em utilizar TICs para auxiliar na gerência de assuntos públicos, tornando possível, assim, melhorar e automatizar muitos serviços em uma cidade. Dentre os serviços que podem ser oferecidos estão os serviços de emergência inteligentes e os serviços de transporte ao passageiro. Nos serviços de emergência inteligentes um evento corresponde à ocorrência de uma situação de emergência (p. e., acidente, incêndio, afogamento). Em um serviço

de transporte, um evento é relacionado a um chamado para levar um passageiro a um local de destino. Como administrar esses eventos é uma questão a ser abordada. Além disso, a questão do atraso deve ser considerada, uma vez que para algumas aplicações em Cidades Inteligentes, como aplicações de saúde e de emergência, atrasos devem ser minimizados. Neste contexto, arquiteturas distribuídas usando estratégias de *Fog Computing* (Computação na borda da rede) poderiam ajudar a minimizar o atraso no envio de eventos, bem como limitar o envio de mensagens para um serviço de *Cloud*. *Fog Computing* é uma plataforma virtualizada que oferece processamento, armazenamento e serviços de rede entre os objetos inteligentes e os tradicionais *Data Centers* de Computação em Nuvem (*Cloud Computing Data Centers*) normalmente, mas não exclusivamente, localizados nas bordas da rede (BONOMI et al., 2012).

Assim, um *middleware publish-subscribe* administrando os eventos enviados entre essas aplicações, as quais se comunicam por meio de eventos, poderia ser utilizado em Cidades Inteligentes juntamente com uma arquitetura distribuída que utiliza *Fog Computing*. Uma rede de objetos inteligentes pode ser formada nesse cenário e um *middleware* ofereceria serviços comuns para as aplicações e facilitaria seu desenvolvimento, especialmente por gerenciar o envio de eventos na rede formada. *Middlewares publish-subscribe*, conhecidos como *middlewares* baseados em evento, serviços de eventos ou ainda serviços de notificação de eventos [1], são apropriados para Cidades Inteligentes. O paradigma *publish-subscribe* é frequentemente utilizado como um mecanismo de comunicação em aplicações baseadas em eventos e está difundindo devido à expansão dos serviços e aplicações de IoT (ANTONIĆ et al., 2015).

Intuitivamente, um serviço de notificação de eventos fornece um mecanismo de seleção e de entrega de eventos. No entanto, durante a entrega de um evento pode ser necessário que somente determinado subconjunto de assinantes, os mais adequados segundo algum critério específico (p. e., localização), recebam aquele evento da maneira mais rápida possível. Um assinante pode apresentar melhores condições para receber o evento e tratá-lo que os outros. Pode-se supor que o assinante possui uma força magnética que permite atrair mensagens para si (CALSAVARA; JR, 2010). Em um serviço de notificação de eventos essa possibilidade de notificação de eventos a um subconjunto de assinantes dentre todos os que se inscreveram para o evento não era possível, conforme verificado nos trabalhos relacionados (Tabela 1). Essa seleção dos melhores assinantes é muito importante em Cidades Inteligentes, especialmente em serviços de emergência a catástrofes naturais ou provocadas pelo homem, em que a tomada de decisão humana é difícil.

[1] Os termos *middleware publish-subscribe* e serviço de notificação de eventos serão utilizados de forma intercalada no texto.

1.1 Objetivo

O objetivo geral do trabalho é a especificação de um novo modelo de eventos (i. e., modelo de comunicação) para *middlewares publish-subscribe* aplicado a serviços em Cidades Inteligentes. O novo modelo chamado *Event to Most Suitable Subscribers (EMSS)* permite o envio de eventos por aptidão de assinante, ou seja, o envio de eventos a um subconjunto dos assinantes mais adequados ou melhores. Os objetivos específicos deste trabalho são apresentados como segue:

1. Propor um *middleware publish-subscribe* distribuído para validar o modelo de eventos proposto.
2. Implementar o *middleware*.
3. Avaliar o modelo e o *middleware*, por meio da simulação dos serviços policial, bombeiro e transporte de passageiros em um ambiente simulado de Cidades Inteligentes, focando especialmente em provar ou refutar a hipótese de pesquisa delineada para o trabalho.

1.2 Hipótese de Pesquisa

Neste trabalho, o estudo foi planejado e conduzido para testar a seguinte hipótese:

- Se for utilizado o modelo de comunicação de eventos por aptidão de assinante proposto, utilizando uma infraestrutura fixa e assinantes móveis geograficamente distribuídos em uma cidade, então haverá ganho de tempo no atendimento aos eventos em relação a uma escolha entre uma porcentagem dos melhores assinantes disponíveis.

1.3 Justificativa e Problematização

Em uma Cidade Inteligente, um conjunto de usuários pode produzir ou consumir informações para a oferta de serviços inteligentes visando melhorar a qualidade de vida dos cidadãos. Usuários podem utilizar um serviço de notificação de eventos para a troca de informações. Desta forma, um usuário pode ser um produtor de eventos (*publisher*) ou um consumidor de eventos (*subscriber*). Cada evento é publicado em um espaço de eventos, o próprio serviço de notificação de eventos, e cada assinante seleciona os eventos que quer receber emitindo subscrições. Cada subscrição pode ser realizada de diversas formas, como por tópico, conteúdo, tipo, conceito ou ainda adaptável a localização. Se a subscrição corresponde ao evento, ou seja, se a correspondência der positivo, somente os assinantes que se inscreveram para aquele evento o recebem (filtragem no lado produtor).

Esse processo se chama de *matching*, uma espécie de filtragem, e é realizado normalmente antes do roteamento do evento, que consiste na entrega daquele evento.

Considerando-se um serviço de notificação de eventos de propósito específico, em que aplicações necessitem do envio de notificações de eventos apenas para um subconjunto dos assinantes mais adequados, como no caso de serviços de cuidados médicos, situações de emergência e transporte de passageiros em Cidades Inteligentes, o roteamento de eventos deve ser diferenciado. Por sua vez, como em todo o serviço de notificação de eventos, as notificações de eventos não são encaminhadas da mesma forma como normalmente acontece com as mensagens que são enviadas pela Internet. Na Internet, as mensagens são enviadas com base no endereço IP (i. e., *Internet Protocol*) do destino, enquanto em um serviço de notificação de eventos é o assinante que determina a entrega das mensagens, não o publicador.

Dentro deste contexto, vários *middlewares* clássicos encontrados na literatura foram analisados para verificar principalmente como acontece a filtragem e o encaminhamento de eventos. Todos eles implementam o clássico modelo de comunicação *publish-subscribe*. Desta forma, os *middlewares* analisados como Siena (CARZANIGA; ROSENBLUM; WOLF, 2001), Hermes (PIETZUCH; BACON, 2002), Steam (MEIER; CAHILL, 2002), SensorBus (RIBEIRO et al., 2005), EMMA (MUSOLESI; MASCOLO; HAILES, 2006), Mires (SOUTO et al., 2006), RUNES (COSTA et al., 2007), PSWare (LAI; CAO; ZHENG, 2009), TinyDDS (BOONMA; SUZUKI, 2012), PRISMA (SILVA et al., 2014), Publish/Subscribe Notification Middleware for Vehicular networks (LEONTIADIS, 2007), Apache Kafka (KAFKA, 2018) e RabbitMQ (RICHARDSON et al., 2018) não asseguram que somente um subconjunto de assinantes, especialmente os mais adequados no contexto do serviço implementado, sejam notificados. A Tabela 1 sumariza esses *middlewares*, na qual as principais características são indicadas. A tabela descreve o tipo de arquitetura da rede de participantes do serviço de eventos, a estratégia de roteamento e como o *middleware* realiza a filtragem dos eventos.

Quanto à arquitetura, o *middleware* precisa ter uma arquitetura distribuída em Cidades Inteligentes, como apontado por (ANTONIĆ et al., 2015). Isso torna possível a redução da carga da rede, favorece a disponibilidade (tolerância a falta) e aumenta a escalabilidade do sistema. Essa arquitetura não apresenta um ponto de falha e nenhum *overhead* de processamento, armazenamento e comunicação. Além de que, em uma situação de emergência como um desastre, por exemplo, se uma arquitetura centralizada fosse adotada a cidade poderia ficar isolada sem receber os serviços de emergência necessários. A arquitetura *peer-to-peer* é apropriada para disseminação de eventos entre um número limitado de participantes e para implantação em pequena escala devido às questões de latência e alto *throughtput* de eventos. Como consequência, *middlewares* com arquitetura centralizada ou *peer-to-peer* não são apropriados para Cidades Inteligentes.

Além disso, em um serviço de eventos, a forma de envio de um evento para

Tabela 1 – Principais *Middlewares Publish-Subscribe*.

Middleware	**Arquitetura**	**Roteamento**	**Filtragem de Eventos**
SIENA (CARZANIGA; ROSENBLUM; WOLF, 2001)	Distribuída	Conteúdo	Subconjunto de participantes
Hermes (PIETZUCH; BACON, 2002)	Distribuída	Rendezvous	Subconjunto de participantes
Steam (MEIER; CAHILL, 2002)	*Peer-to-peer*	Epidêmico	Publicador, Assinante
SensorBus (RIBEIRO et al., 2005)	Centralizada	*Pull*	Assinante
EMMA (MUSOLESI; MASCOLO; HAILES, 2006)	*Peer-to-peer*	Epidêmico	Publicador, Assinante
Mires (SOUTO et al., 2006)	Distribuída	*Multihop*	Publicador
PSNMVN (LEONTIADIS, 2007)	Distribuída	*Event Flooding*	Assinante
RUNES (COSTA et al., 2007)	Distribuída	*Multihop*	Publicador
PSWare (LAI; CAO; ZHENG, 2009)	Distribuída	*Multihop*	Publicador
TinnyDDS (BOONMA; SUZUKI, 2012)	Distribuída	DHT/ST*	Publicador
PRISMA (SILVA et al., 2014)	Centralizada	*Pull*	Assinante
Apache Kafka (KAFKA, 2018)	Distribuída	Tópico	Subconjunto de participantes
RabbitMQ (PIVOTAL, 2019)	Distribuída	Tópico	Subconjunto de participantes

*DHT - *Distributed Hash Table*; ST - *Spanning Tree*.

Fonte: Autoria Própria

um subconjunto de assinantes, os mais apropriados, em um conjunto de todos os que receberiam o evento em uma Cidade Inteligente é uma questão importante a ser abordada. Primeiramente, quando ocorre o *matching* em um serviço de eventos, o evento segue para o assinante. Em uma subscrição baseada em tópicos, quando um evento do tópico ocorre ele segue normalmente para a notificação do assinante. É como se existisse um canal ligando publicador e assinante. A identificação do assinante na subscrição é essencial para os algoritmos de roteamento, pois ajuda na notificação do evento. Embora seja utilizada, a identificação do assinante original é modificada enquanto a subscrição flui pela rede de participantes. Consequentemente, não tem como saber que assinante realmente recebeu o evento e quantos receberam, especialmente nos algoritmos de roteamento *event flooding* e baseados em filtro ou conteúdo. Isso porque muitas vezes quem recebe o evento encaminha para o vizinho e assim sucessivamente até alcançar o assinante ou ainda a subscrição permanece com o assinante não sendo enviada para a rede de participantes, como no caso do algoritmo *event flooding*. A única exceção se aplica ao algoritmo *subscription flooding*, em que a subscrição é encaminhada para todos os participantes vizinhos com a identificação direta do assinante sem modificação alguma em seu conteúdo pelos participantes. Esse primeiro indício mostra que o envio a um subconjunto de interessados dentro de um

conjunto de todos os que receberiam o evento não é uma tarefa trivial.

Segundo, os filtros nas subscrições seriam um problema, especialmente se o *matching* dependesse de um comportamento observável do assinante e de outros atributos de difícil mensuração. A definição de alguns parâmetros de comparação poderia implicar, dependendo do contexto, ao não recebimento do evento pelos melhores assinantes. Para melhor entendimento, imagine uma cidade em que veículos de resgate tem que se movimentar para atender a ocorrências de emergência. Eles podem se movimentar pela cidade a maior parte do tempo ou então ficar em estações base e se deslocarem somente quando são solicitados para tal. A filtragem no lado do consumidor é inviável neste caso, pois muitos veículos seriam acionados comprometendo o funcionamento do sistema de eventos como um todo, independente do modelo de subscrição adotado. Logo, no caso de situações de emergência e desastres, este tipo de filtragem não é apropriado pois mais assinantes provavelmente seriam acionados.

Quanto à filtragem no lado produtor, quando uma subscrição fosse realizada, possivelmente, um parâmetro associado à distância seria informado e como identificar quais os assinantes estão mais próximos ou mais distantes do evento seria um problema. A questão diz respeito à faixa de valores para a distância que deveria ser adotada no assinante para a especificação de filtros na subscrição, e mesmo no publicador para a fase de *matching*. O problema se agrava se um critério de nível de observação do comportamento do assinante fosse adotado pelo publicador em relação ao assinante como, por exemplo, o nível de cansaço da equipe de atendimento da emergência. De uma forma geral, o estado das subscrições e dos eventos é distribuído na rede de participantes que é utilizada pelo serviço de eventos e como estimar valores de comparação nas subscrições (filtros) e no *matching* para que a precisão do serviço de notificação de eventos não fique comprometida é uma questão delicada. Dependendo dos valores utilizados, emergências poderiam ficar sem atendimento devido a restrições impostas por seus assinantes (i. e., filtros nas subscrições) quando os mesmos poderiam e deveriam ser notificados das emergências. Além disso, em um cenário dinâmico, no qual unidades móveis permanecem se movimentando durante o tempo, as coordenadas ou mesmo a distância informada na subscrição estaria desatualizada no momento do *matching* para o envio do evento. Sendo assim, a subscrição precisa ser atualizada de alguma forma.

Diante do exposto, o que foi proposto para a resolução dos problemas acima mencionados é a realização de um cálculo sobre os valores informados nas subscrições (i. e., um valor de aptidão ou *fitness*) e sua constante atualização, pois a dinâmica do ambiente é levada em conta. Por consequente, esse valor de aptidão é calculado para cada assinante, o qual permite a seleção proposta neste trabalho. Cada valor significa o quão um assinante está preparado para tratar um evento. Assim, com o cálculo tem-se métricas para o envio de eventos para os assinantes mais adequados, ou seja, para especificação de

um modelo de comunicação de eventos com filtragem de eventos por aptidão de assinante. A decisão de quais assinantes, ou seja, os mais aptos que devem receber o evento, fica sob a responsabilidade do *middleware* que implementa o modelo de eventos. Essa abordagem difere dos filtros de eventos, nos quais as várias assinaturas são comparadas de acordo com um conjunto de regras para obter uma correspondência antes de enviar o evento (abordagem de filtragem do lado do servidor). Tal correspondência não garante que um subconjunto de assinantes receba o evento, mas que todos os assinantes na correspondência o recebam. O cálculo sobre as assinaturas e o comportamento observado dos assinantes não foi encontrado nos trabalhos relacionados. Com o evento publicado, o *matching* no EMSS compara o tópico do evento gerado com o tópico da subscrição, primeira filtragem, e depois utiliza o valor de aptidão para escolher dentre os valores disponíveis os mais aptos e necessários, ou seja, os que possuem maior valor em número necessário, o que equivale a uma filtragem secundária. Por consequente, o envio do evento somente aos assinantes mais apropriados é realizado. A análise de recursos computacionais necessários para a execução dos algoritmos de cálculo dos valores de *fitness* está fora do escopo deste trabalho.

Desta forma, se for considerado um cenário de acidentes de trânsito em que existe mais de um hospital para o recebimento dos feridos, o envio do evento poderá ser para alguns dos hospitais que se encontrarem mais próximos ao local do acidente ou então de acordo com as condições desses hospitais em receber os pacientes (p. e., número de leitos, equipamentos apropriados para exame, se possui Unidade de Terapia Intensiva). Logo, o evento chegará somente aos hospitais com melhores condições, não a todos, fazendo com que somente os médicos e enfermeiros dos hospitais selecionados se mobilizem para o atendimento. Também no caso do SAMU ou Corpo de Bombeiros, várias ambulâncias, por exemplo, poderiam ser deslocadas até o acidente em questão sem necessidade. A seleção das ambulâncias mais apropriadas evitaria isso. Em um serviço policial, somente algumas viaturas atenderiam a um chamado impedindo, assim, que várias outras se movessem até o local. Ou então, de um grande conjunto de viaturas, somente uma poderia ser acionada. Com isso, ter-se-ia redução de custos, minimização do tempo de chegada para resolver um chamado, agilidade no atendimento, evitaria deslocamentos desnecessários, falhas de comunicação, entre outros. Em outras palavras, os prejuízos humanos e materiais seriam minimizados. O processo de seleção dos melhores assinantes e notificação de eventos é realizado pelo *middleware*, ficando somente para os publicadores a tarefa de publicar os eventos e para os assinantes, que compõe o *middleware*, a tarefa de se inscrever-se para recebê-los. Os eventos fluem por uma rede de servidores que possui como finalidade o encaminhamento dos eventos de acordo com as regras de comunicação estabelecidas pelo modelo de comunicação por aptidão de assinante EMSS. O modelo de eventos EMSS procura dar um comportamento humano a um protocolo baseado em eventos a ser utilizado em aplicações que necessitem do envio seletivo de eventos em Cidades Inteligentes.

1.4 Contribuição

A principal contribuição do trabalho é um novo modelo de comunicação baseado em eventos com filtragem de eventos por aptidão de assinante chamado EMSS a ser utilizado em *middlewares publish-subscribe* no contexto de Cidades Inteligentes. O modelo de comunicação pode ser entendido como um protocolo baseado em eventos, protocolo da camada de aplicação, para o encaminhamento de eventos de forma seletiva. No contexto de *middlewares publish-subscribe*, o protocolo baseado em eventos é o modelo de eventos. Um *middleware* de mesmo nome foi implementado para validação do modelo. Logo, o *middleware* faz parte da contribuição principal deste trabalho. Assim, o trabalho inclui um *middleware publish-subscribe* e o modelo de eventos baseado em filtragem de eventos por atptidão de assinantes EMSS. Desta forma, o modelo de eventos EMSS e o *middleware* que o implementa são os diferenciais deste trabalho.

1.5 Publicações

Os principais resultados relatados nesta tese são descritos nas seguintes publicações:

- HERNANDES, Sediane C. L. et al. A New Event Model for Event Notification Services Applied to Transport Services in Smart Cities. In: **International Conference on Information Networking (ICOIN 2020)**. Barcelona, Espanha, 2020.
- HERNANDES, Sediane C. L. et al. Um Estudo sobre Middlewares Publish-Subscribe para Envio Seletivo de Eventos em Cidades Inteligentes. In: **Conferência Latino-americana de Informática (CLEI 2019)**. Cidade do Panamá, Panamá, 2019.
- HERNANDES, Sediane C. L. et al. An Efficient Event-Based Protocol for Emergency Situations in Smart Cities. In: **International Conference on Advanced Information Networking and Applications (AINA 2019)**. Springer, Cham, 2019. p. 523-534.
- HERNANDES, Sediane Carmem Lunardi; PELLENZ, Marcelo Eduardo; CALSAVARA, Alcides. An Architecture of Fog Computing in Smart Cities: the middleware E2BS in emergency calls. In: **XLIV Latin American Computer Conference (CLEI 2018)**. IEEE, 2018. p. 509-518.
- HERNANDES, Sediane C. L.; PELLENZ, Marcelo Eduardo; CALSAVARA, Alcides. O Middleware E2BS Aplicado ao Acionamento de Unidades Móveis em Cidades Inteligentes para Situações de Emergência. In: **Anais do II Workshop de Computação Urbana (COURB 2018)**. SBRC, 2018.

- HERNANDES, Sediane C. L.; CALSAVARA, Alcides; LIMA JR, Luiz AP. Acionamento Inteligente de Unidades Móveis em Situações de Emergência em Cidades. In: **Anais do I Workshop de Computação Urbana (COURB 2017)**. SBRC, 2017. p. 58-71.

- HERNANDES, Sediane C. L. et al. Serviços de Emergência em Cidades Inteligentes: o Problema de Acionamento de Unidades Móveis. **Revista Eletrônica de Sistemas de Informaçao (RESI)**, v. 16, n. 2, p. 1-31, 2017.

1.6 Suporte Financeiro

Esta tese teve o apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) através do Programa de Suporte à Pós-Graduação das Instituições de Ensino Particular (PROSUP) no âmbito do Código 001, posteriormente substituído pelo Programa de Apoio à Pós-Graduação em Instituições Comunitárias de Ensino Superior (PROSUC).

1.7 Divisão do Trabalho

O trabalho está dividido como segue. O capítulo 2 apresenta os conceitos de Cidades Inteligentes. Serviços de notificação de eventos são descritos no capítulo 3. O capítulo 4 descreve o modelo de comunicação de eventos EMSS e sua implementação por meio de um *middleware* de mesmo nome. O capítulo 5 apresenta e discute os resultados computacionais simulados. Algumas considerações finais são apresentadas no capítulo 6.

2 Cidades Inteligentes

Apesar de existir um aumento na frequência do uso do termo Cidade Inteligente, uma definição concreta de uma Cidade Inteligente ainda está emergindo e várias definições têm sido dadas com diferentes pontos de vista (YIN et al., 2015); entretanto, convergindo para um ponto em comum, o aumento da qualidade de vida dos cidadãos. Em termos genéricos Cidade Inteligente é um ambiente urbano que utiliza Tecnologias de Informação e Comunicação (TICs) e outras tecnologias relacionadas para aumentar a eficiência e a qualidade dos serviços das operações regulares da cidade oferecida aos cidadãos urbanos (SILVA; KHAN; HAN, 2018). Uma definição mais formal pode ser encontrada em (ITU-T Focus Group on Smart Sustainable Cities, 2014), a qual apresenta uma Cidade Sustentável Inteligente (*Smart Sustainable City - SSC*) como uma cidade que usa as TCIs e outros meios para melhorar a qualidade de vida dos cidadãos, a eficiência dos serviços e operações urbanas e a competitividade, enquanto assegura que ela atenda às necessidades das gerações presente e futura em relação a aspectos econômicos, sociais e ambientais. A qualidade de vida pode ser medida em termos de bem-estar financeiro e emocional de seus habitantes. Para (MOHANTY; CHOPPALI; KOUGIANOS, 2016), em uma Cidade Inteligente, as tecnologias digitais traduzem-se na melhoria dos serviços públicos para os habitantes e melhor uso dos recursos enquanto reduz os impactos ambientais. Por outro lado, (SILVA; KHAN; HAN, 2018) menciona que uma Cidade Inteligente é uma aplicação da noção de Internet das Coisas, uma vez que herda os mecanismos operacionais (subjacentes) de Internet das Coisas.

A Internet das coisas, também chamada de Internet de tudo ou Internet industrial, é um novo paradigma tecnológico em que uma rede global de máquinas e dispositivos (objetos inteligentes ou coisas) é formada, em que esses objetos inteligentes são capazes de interagir uns com os outros (LEE; LEE, 2015). Um objeto inteligente é um objeto físico ou digital com capacidade de sensoriamento, processamento e rede (KORTUEM et al., 2009), além de capacidade de memória (LOUREIRO et al., 2003). Esses objetos carregam parte da lógica da aplicação, podendo "sentir", registrar e interpretar o que está ocorrendo em si mesmos e no mundo, agir por conta própria, se intercomunicando uns com os outros e trocando informações com as pessoas (KORTUEM et al., 2009). A geração, troca e consumo de dados deve acontecer com o mínimo de interação humana (ROSE; ELDRIDGE; CHAPIN, 2015). Para (KOPETZ, 2011), um objeto inteligente é apenas outro nome dado para um sistema embutido que está conectado à Internet. Objetos inteligentes podem ser sensores, elementos atuadores, objetos que possuem um chip RFID e também outras tecnologias, como por exemplo, eletrodomésticos, câmeras de segurança, veículos, entre outros.

Além disso, os objetos inteligentes conectados à Internet podem ser controlados remotamente e ser provedores e consumidores de serviços de/e para outros objetos ou aplicações, prover dados para análise e reagir a eventos. Tudo isso sendo possível porque são unicamente endereçados (BASSI; HORN,). Em resumo, IoT transforma estes objetos físicos de tradicionais em inteligentes, explorando suas tecnologias subjacentes, como computação ubíqua e disseminada, dispositivos embutidos, tecnologias de comunicação, redes de sensores, protocolos da Internet e aplicações. O núcleo da implementação de Cidades Inteligentes é a Internet das Coisas, ou seja, IoT é o *backbone* de Cidades Inteligentes (MOHANTY; CHOPPALI; KOUGIANOS, 2016).

Cidades Inteligentes e IoT possuem ideologias similares, mas com diferentes origens. Cidades Inteligentes surgiram para resolver os problemas das cidades modernas ou grandes cidades, enquanto IoT surgiu inicialmente para conectar fisicamente dispositivos a Internet (VERMESAN et al., 2011). IoT é conduzida por avanços tecnológicos, não por aplicações ou necessidades dos usuários. IoT vai em direção às necessidades dos usuários quando é utilizada em um contexto específico, como a gerência do lixo em uma grande cidade, por exemplo. Em resumo, Internet das Coisas e Cidades Inteligentes são conceitos estreitamente relacionados. Cidades Inteligentes não existiriam sem IoT e, da mesma forma, IoT não teria sua devida importância sem seu uso em um contexto urbano.

2.1 Internet das Coisas como Infraestrutura de Cidades Inteligentes

A Internet das Coisas como *backbone* em Cidades Inteligentes permite que objetos inteligentes se conectem e se comuniquem. Em Internet das Coisas, uma arquitetura poderia ser definida como um sistema descentralizado, fracamente acoplado, de objetos inteligentes. Conhecer as características deste objetos e as formas com que podem se comunicar é muito importante.

2.1.1 Características de IoT

As principais características de IoT são apresentadas em (RAZZAQUE et al., 2016), que são:

- **Dispositivos heterogêneos:** a diversidade de dispositivos de IoT (p. e., smartphones, laptops, sensores e atuadores, tags RFID) surge por causa das diferentes capacidades e características dos próprios dispositivos que fazem parte dos componentes de IoT, bem como outras razões como produtos de vários fabricantes e os requisitos das aplicações.
- **Recursos limitados:** os dispositivos que fazem parte de IoT possuem limites em relação à sua capacidade de processamento, memória e capacidade de comunicação.

Os dispositivos RFID ou tags RFIC/NFC, por exemplo, não possuem qualquer capacidade de processamento ou bateria para energizá-las, ao contrário de outros dispositivos como celulares, por exemplo.

- **Interação espontânea:** em aplicações IoT, as interações entre objetos podem acontecer quando um objeto entra na faixa de comunicação de outro(s) objeto(s), levando à geração espontânea de eventos. Por exemplo, um smartphone pode vir a entrar em contato com uma TV ou refrigerador ou uma máquina de lavar e gerar eventos sem o envolvimento do usuário. Normalmente, em IoT, uma interação com um objeto significa que um evento é gerado e é empurrado para o sistema sem muita atenção humana.

- **Rede em grande escala e grande número de eventos:** em IoT milhares de objetos inteligentes podem interagir em uma escala muito maior do que os sistemas em rede tradicionais. A interação espontânea de eventos entre um grande número de objetos pode produzir uma enorme quantidade de eventos. Esse número não controlado de eventos pode causar problemas de congestionamento e reduzir a capacidade de processamento de eventos dos próprios objetos e da rede de comunicação.

- **Rede dinâmica e sem infraestrutura:** IoT pode integrar dispositivos, muitos dos quais podem ser móveis, com conexão sem fio e com recursos limitados. Nós móveis dentro da rede saem ou se unem à rede a qualquer hora. Além disso, os nós podem ser desconectados devido a uma conexão sem fio ruim ou falta de bateria. Isso faz de IoT um ambiente altamente dinâmico. Dentro desse ambiente *ad-hoc*[1], no qual há pouca ou nenhuma conexão com uma infraestrutura fixa, é difícil manter uma rede estável para suportar vários cenários de aplicação que dependem de IoT. Cidadãos podem cooperar para manter a rede conectada e ativa.

- **Adaptável ao contexto:** contexto é chave em IoT e suas aplicações. Um grande número de sensores pode gerar uma grande quantidade de dados que sem análise não possuem qualquer valor. Computação adaptável ao contexto armazena informação de contexto relacionada aos dados do sensor, facilitando assim a interpretação desses dados.

- **Inteligência:** objetos inteligentes poderão interoperar e agir independentemente baseados no contexto, nas circunstâncias ou ambientes.

- **Adaptável à localização:** a localização sobre objetos ou sensores em Internet das Coisas é crítica, desempenhando um papel importante em computação adaptável ao contexto. Em uma rede de grande escala de coisas, as interações são altamente

[1] Redes *ad-hoc* são definidas pela maneira pela qual os nós da rede são organizados (de forma improvisada) para fornecer caminhos para os dados serem encaminhados do usuário para e do destino desejado. A rede é auto-configurável. Redes de sensores sem fio são uma classe das redes *ad-hoc*.

dependentes de seus lugares, arredores e a presença de outras entidades (p. e., coisas e pessoas).

- **Distribuída:** a rede IoT pode possuir diferentes escalas, ou seja, uma escala global como a Internet ou local dentro de uma área de aplicação.

Conhecer as características do ambiente IoT auxilia no desenvolvimento de aplicações, pois muitas vezes as características devem ser consideradas em seu projeto, bem como o ambiente no qual a aplicação IoT é utilizada. Assim, neste trabalho IoT será utilizada em um contexto urbano, a saber em Cidades Inteligentes. No contexto delineado, os objetos terão uma infraestrutura fixa de comunicação o que faz com que a rede seja estável, embora este objetos poderão entrar e sair dessa rede a qualquer hora.

2.1.2 Padrões de Comunicação para Objetos Inteligentes em IoT

A comunicação entre objetos inteligentes pode seguir vários padrões ou modelos, entre os quais (TSCHOFENIG et al., 2015), (ARKKO; THALER; MCPHERSON, 2015):

2.1.2.1 Padrão de Comunicação Device-to-Device

Esse modelo de comunicação permite que dois ou mais dispositivos se conectem diretamente e se comuniquem entre si, não necessitando, assim, de um servidor de aplicação intermediário. Esses dispositivos podem comunicar-se sobre vários tipos de redes, incluindo redes IP (p. e., a Internet) ou então utilizando protocolos como *Bluetooth*, *ZWave* ou *ZigBee* para estabelecer comunicação direta dispositivo a dispositivo através de comunicação sem fio.

2.1.2.2 Padrão de Comunicação Device-to-Cloud

Nesse modelo de comunicação, os dispositivos se conectam diretamente a um serviço de nuvem na Internet, semelhante a um provedor de serviço de aplicação para troca de dados e controle do tráfego das mensagens. Essa abordagem possui como vantagem o fato da existência de um mecanismo de comunicação como Ethernet com fio ou conexões Wi-fi para estabelecer uma conexão entre o dispositivo e a rede IP, a qual se conecta, posteriormente, a um serviço de nuvem. Esse modelo de comunicação é utilizado pela SmartTV da Samsung (ARKKO; THALER; MCPHERSON, 2015), a qual coleta, utiliza, compartilha e armazena informações relacionadas ao usuário em relação ao uso da TV. Essas informações são enviadas por meio da Internet e são utilizadas para, por exemplo, melhorar serviços da TV na Samsung, como o serviço de reconhecimento facial e controle de gestos, recomendar conteúdos, aplicativos ou programas ao usuário com base em suas preferências e hábitos e no caso de programas ao vivo mostrar os disponíveis na região do usuário com base em seu código postal.

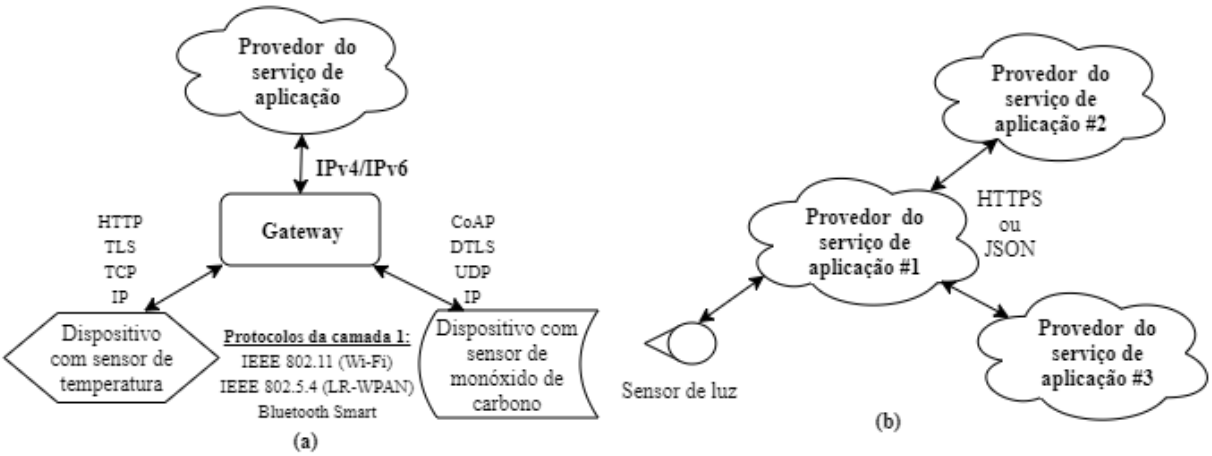
2.1.2.3 Padrão de Comunicação Device-to-Gateway

Nesse modelo, o dispositivo IoT conecta-se através de um serviço de *gateway* em nível de aplicação como uma forma para alcançar o serviço de nuvem (Figura 1 (a)). Em muitos casos, o *gateway* é um smartphone executando uma aplicação para comunicar-se com um dispositivo e retransmitir os dados coletados pelo dispositivo para um serviço de nuvem. Um exemplo é um aplicativo de ginástica pessoal que precisa de um smartphone agindo como um *gateway* para conectar o dispositivo de ginástica à nuvem.

2.1.2.4 Padrão de Comunicação Back-End Data Sharing

No padrão de comunicação *Device-to-Cloud*, os dispositivos IoT enviam dados somente a um único provedor de serviço de aplicação. Entretanto, muitas vezes o usuário precisa exportar e analisar dados em conjunto com outras fontes de dados. Essa abordagem é uma extensão do padrão de comunicação *Device-to-Cloud*, na qual dados podem ser enviados e recebidos de várias fontes de dados. A Figura 1 (b) ilustra um exemplo de aplicação deste padrão de comunicação.

Figura 1 – (a) Padrão de comunicação *Device-to-Gateway* (b) Padrão de comunicação *Back-end Data Sharing*.



Fonte: (TSCHOFENIG et al., 2015), (ARKKO; THALER; MCPHERSON, 2015)

Salienta-se que mais de um padrão de comunicação pode ser aplicado ao mesmo tempo em um objeto inteligente.

2.2 Arquiteturas para Cidades Inteligentes

Várias arquiteturas estão sendo propostas para IoT, as quais podem ser utilizadas em um contexto urbano. Entretanto, não existe uma arquitetura padronizada ainda. Em (KHAN et al., 2012), uma arquitetura genérica para IoT em 5 camadas é apresentada, sendo composta das camadas de percepção, rede, *middleware*, aplicação e negócios. A camada de percepção é conhecida como camada de dispositivos, consiste de objetos físicos (p. e., tags RFID, código de barras, sensores infravermelhos) e sensores (p. e., localização, temperatura, umidade, vibração) e possui como função tratar da identificação e coleta de informações especificas dos objetos físicos e sensores. A informação coletada é passada para a camada de rede. A camada de rede, também conhecida como camada de transmissão, tem por função transferir com segurança a informação dos dispositivos físicos e sensores para o sistema de processamento da informação. Os meios de transmissão podem ser diversos com fio ou sem fio (p. e., 3G, 4G, *Wifi*, *Bluetooth*, *ZigBee*). Desta forma, a camada de rede transfere a informação da camada de percepção para a camada do *middleware*, sendo esta a responsável pelo gerenciamento de serviços. O *middleware* recebe as informações da camada de rede e pode armazenar em um banco de dados, realizar o processamento de informações, tomar decisões automáticas com base nos resultados, entre outros. A camada de aplicação oferece gerenciamento global das aplicações baseada nos objetos de informação processados na camada do *middleware*. Por fim, a camada de negócios é responsável pela gerência do sistema IoT ajudando a determinar ações futuras e estratégias de negócios.

Em (SILVA; KHAN; HAN, 2018) é proposta uma arquitetura genérica para Cidades Inteligentes, a qual é dividida em 4 camadas: sensoriamento (coleta de dados), transmissão, gerência de dados e aplicação. Na camada de coleta de dados, encontra-se os dispositivos físicos e infraestrutura, redes de sensores sem fio, entre outros. Para conectar as fontes de dados com as estações de gerência, a camada de transmissão age como um *backbone* da arquitetura. A camada de transmissão é a convergência de várias redes de comunicação e consiste de várias tecnologias com fio, sem fio e satélite. Essa camada é dividida em camada de acesso à rede e camada de transmissão de rede. Como tecnologias de acesso a rede que oferecem cobertura de curto alcance estão o *Bluetooth*, *ZigBee*, *Near Field Communication (NFC)*, RFID e Zwave. Como exemplo de tecnologias de transmissão de redes com cobertura mais ampla estão a 3G, 4G *Long-Term Evolution (LTE)*, 5G e *Low-Power Wide Area Networks (LP-WAN)*. A camada de gerência de dados pode ser caracterizada pela fusão, análise, processamento, armazenamento de dados e gerência de decisão e eventos. A gerência de eventos pode ser realizada por um *middleware*, pois muitos eventos são gerados pelas aplicações IoT e devem ser gerenciados como parte desta camada (RAZZAQUE et al., 2016). A camada de aplicação, por outro lado, executa decisões sobre dados recebidos da camada de gerência de dados. É a camada que interage com o usuário e contém as aplicações para Cidades Inteligentes.

Em (YANG et al., 2011) é proposta uma arquitetura para IoT em 3 camadas (percepção, rede e aplicação). A camada de percepção possui como função coletar as informações dos objetos inteligentes, redes de sensores sem fio, entre outros. A camada de rede possui como função transmitir a informação a uma longa distância, sendo formada de várias redes. Além disso, possui como função o processamento inteligente das informações. Dessa forma, a camada de rede compreende as camadas de transmissão e gerência de dados da arquitetura encontrada em (SILVA; KHAN; HAN, 2018). A camada de aplicação fornece um serviço específico aos usuários com base nos dados.

As arquiteturas apresentadas, basicamente, possuem camadas com funções equivalentes. A camada de percepção das arquiteturas (KHAN et al., 2012) e (YANG et al., 2011) possui a mesma função da arquitetura apresentada em (SILVA; KHAN; HAN, 2018). A camada de rede apresentada em (KHAN et al., 2012) possui função equivalente a camada de transmissão proposta em (SILVA; KHAN; HAN, 2018), bem como uma das funções da camada de rede descrita em (YANG et al., 2011). A camada de *middleware* (KHAN et al., 2012) possui funções equivalentes a camada de gerência de dados de (SILVA; KHAN; HAN, 2018) e de uma das funções da camada de rede de (YANG et al., 2011). A camada de aplicação das arquiteturas apresentadas, possuem basicamente as mesmas funções, exceto em (KHAN et al., 2012) que a camada de negócios apresenta as funcionalidades da camada de aplicação e a camada de aplicação funções de gerência de dados. Desta forma, a Figura 2 apresenta uma adaptação das arquiteturas (SILVA; KHAN; HAN, 2018), (KHAN et al., 2012) e (YANG et al., 2011) que será utilizada neste trabalho.

Figura 2 – Arquitetura em camadas de uma Cidade Inteligente Genérica.

Fonte: Adaptação de (SILVA; KHAN; HAN, 2018), (KHAN et al., 2012) e (YANG et al., 2011)

2.2.1 Fog Computing como parte da Camada de Transmissão

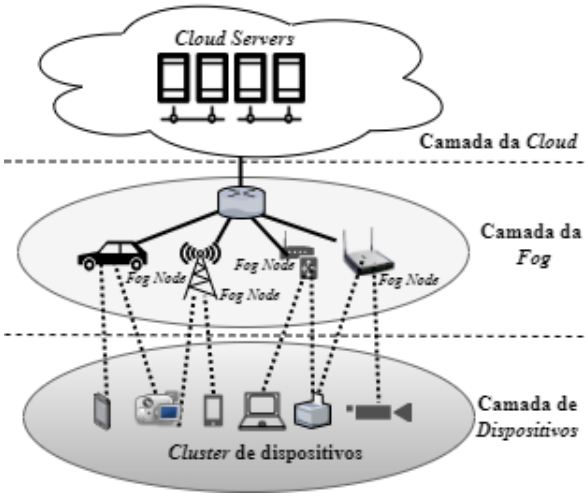
Na camada de transmissão, uma plataforma adicional pode ser utilizada como *Fog Computing*. Essa plataforma é necessária para que alguns requisitos de aplicações IoT (p. e., aplicações de Cidades Inteligentes) sejam conseguidos como baixa atraso na entrega de pacotes, distribuição geográfica, mobilidade, gerência de um grande número de nós, acesso predominantemente sem fio, aplicações de tempo real e heterogeneidade (BONOMI et al., 2012). As camadas de *middleware* e aplicação, desta forma, podem fazer parte de uma camada única, em um dispositivo inteligente na *Fog*.

O advento da Internet das Coisas tem possibilitado a introdução de novas arquiteturas de rede que visam aprimorar o paradigma da Computação em Nuvem atualmente implementado. Objetos do dia a dia, cada um com um identificador único, se conectarão automaticamente a várias redes e farão *upload* de uma grande quantidade de dados de diversos tipos. A infraestrutura da nuvem é incapaz de lidar com o volume, a variedade e a velocidade de dados de IoT, especialmente levando em consideração a latência introduzida ao tentar transferir conjuntos de dados em massa para servidores distantes ou a largura de banda necessária para essas transferências (AKRIVOPOULOS et al., 2017). As redes atuais devem se adaptar para lidar com os requisitos específicos das aplicações de IoT (p. e., baixa latência), uma vez que os recursos podem ser requisitados sob demanda simultaneamente pelos inúmeros dispositivos em diferentes locais (SANTOS et al., 2017). Portanto, faz-se necessário estabelecer um modelo de computação que venha a diminuir essas desvantagens para aplicações IoT como, por exemplo, *Fog Computing*.

Fog Computing é uma plataforma altamente virtualizada que oferece processamento, armazenamento e serviços de rede entre dispositivos finais e os tradicionais *Data Centers* de Computação em Nuvem (*Cloud Computing Data Centers*) normalmente, mas não exclusivamente, localizados nas bordas da rede (BONOMI et al., 2012). *Fog Computing* é um paradigma computacional distribuído especialmente colocado entre os sensores/dispositivos de IoT e os *Cloud DataCenters* (AKRIVOPOULOS et al., 2017). É uma plataforma projetada principalmente para casos de uso de IoT (WANG et al., 2017). *Fog Computing* estende o paradigma tradicional da Computação em Nuvem até a borda (AAZAM; HUH, 2015). Essa ampliação acontece ao migrar o processamento de dados para mais perto do local de produção, acelerando a capacidade de resposta do sistema aos eventos e eliminando a ida e volta dos dados para a Nuvem (AKRIVOPOULOS et al., 2017). A proximidade física da infraestrutura de Nuvem com os sensores ligados a recursos de qualquer aplicativo relacionado à IoT, permite latência limitada, além de menor consumo de largura de banda (AKRIVOPOULOS et al., 2017). O descarregamento de grandes conjuntos de dados para a rede principal não é mais uma necessidade. É um paradigma de *Micro Data Center* (MDC) (AAZAM; HUH, 2015). A arquitetura de *Fog Computing* é mostrada na Figura 3, a qual contém 3 camadas: a camada da Nuvem, a camada *Fog* e a camada dos dispositivos.

A camada da *Fog* pode conter múltiplas camadas de acordo com os requisitos. Os nodos do *Fog* podem ser pequenas estações base, veículos, pontos de acesso WiFi e terminais do usuário. Os dispositivos escolhem o nodo do *Fog* mais apropriado para associarem-se a ele.

Figura 3 – Arquitetura de *Fog Computing*.



Fonte: Adaptação de (WANG et al., 2017)

A distribuição geográfica em que serviços e aplicações na *Fog* demandam implantação amplamente distribuída é uma característica que torna *Fog Computing* uma extensão não trivial da Nuvem, a qual é mais centralizada (BONOMI et al., 2012), (AAZAM; HUH, 2015). O *Fog*, por meio de *proxies* e pontos de acesso posicionados de forma distribuída pela cidade, pode fornecer informações de qualidade para aplicações de IoT, como aplicações para Cidades Inteligentes. O *Fog* fornece entrega de dados em tempo real, especialmente para serviços relacionados à saúde, sensíveis ao atraso e de emergência. Ele pode ajudar os nós com recursos limitados a descarregar tarefas, pré-processar dados brutos e notificar a nuvem, antes que a nuvem possa adaptá-los aos serviços aprimorados. A principal vantagem do *Fog* é suportar a rede na borda, junto com todos os serviços críticos em atraso que podem ser implantados nessa camada (AKRIVOPOULOS et al., 2017). Serviços de emergência ou de monitoramento da saúde são exemplo em que um grande atraso pode impactar significantemente em seu desempenho (SANTOS et al., 2017). Situações em que vida está em perigo requerem baixo atraso e espaços de armazenamento seguro por questões de privacidade (GUIBERT et al., 2017). Situações de emergência e de monitoramento de saúde demandam comunicação eficiente. Para aplicações IoT com restrições de tempo real, a baixa latência pode ser crucial e, além disso, o número de saltos entre o serviço alocado e o dispositivo final deve ser diminuído (SANTOS et al., 2017).

Em relação a tolerância a faltas, *Fog Computing* poderia ajudar, uma vez que o processamento e armazenamento é distribuído nas bordas da rede. Existe uma redução

de pontos de falha e se um ou mais *Fog nodes* falharem, o conteúdo de cada um deve ser recuperado em outro nó. Em situações de desastres naturais, onde parte dos serviços e da rede estariam indisponíveis por diversar razões, alguns *Fog nodes* poderiam manter cópias dos dados de outros *Fog nodes* o que ajudaria a fazer com que as emergências fossem atendidas com a brevidade necessária.

2.3 Aplicações de Cidades Inteligentes

A construção de Cidades Inteligentes envolve o desenvolvimento de aplicação (SU; LI; FU, 2011) para a oferta de serviços. Alguns exemplos de aplicações que oferecem serviços possíveis em Cidades Inteligentes com o uso de IoT no ambiente urbano são (ZANELLA et al., 2014), (SU; LI; FU, 2011): saúde estrutural de construções, gerência do lixo, monitoração da qualidade do ar, monitoração do consumo de energia e do ruído, estacionamento inteligente, monitoramento do congestionamento de tráfego, Redes Inteligentes (*Smart Grids*), Sistemas de Transporte Inteligentes (*Intelligent Transportation System*), Casas Inteligentes (*Smart Home*), Água Inteligente (*Smart Water*), Cuidados Médicos (*Medical Care*), Comida Inteligente (*Smart Food*) e o atendimento a situações de emergência.

Saúde estrutural de construções (*Structural Health of Buildings*) trata da manutenção adequada de construções históricas de uma cidade. A monitoração contínua das condições atuais de cada construção requer a identificação das áreas que são mais sujeitas aos impactos de agentes externos. IoT urbano, neste sentido, pode oferecer uma base de dados distribuída de medições da integridade estrutural das construções. Essa base de dados seria alimentada por dados coletados por sensores, os quais seriam implantados nessas construções. Sensores de vibração e deformação seriam utilizados para monitoramento do stress da construção, sensores atmosféricos nas áreas próximas ajudariam no monitoramento dos níveis de poluição, além de que temperatura e umidade seriam usadas para caracterização completa das condições ambientais. Com isso, o número de testes estruturais periódicos seria reduzido, os quais são caros e realizados por pessoal especializado.

A gerência de lixo implica no uso de lixeiras inteligentes, as quais detectam o nível de carga. Assim, as rotas de coletas dos caminhões de lixo podem ser otimizadas e os custos da coleta de lixo podem ser reduzidos. Em (GUTIERREZ et al., 2015) é apresentado um sistema de coleta de lixo inteligente. O sistema de coleta de lixo inteligente é baseado em dados do nível de lixo de lixeiras em uma área metropolitana. Os dados coletados por sensores são enviados pela Internet para um servidor onde eles são armazenados e processados. Os dados coletados são usados para monitorar e otimizar a seleção diária de lixeiras a serem coletadas, calculando as rotas de acordo com as lixeiras. Todos os dias

os motoristas dos caminhões do lixo recebem novas rotas calculadas em seus dispositivos de comunicação. A característica principal desse sistema é que ele foi projetado para aprender de experiências e tomar decisões não somente do estado do nível de lixo diário, mas também de previsões do estado futuro, congestionamento de tráfego, funções custo-benefício balanceadas e outros fatores que, a priori, os seres humanos não podem prever. A taxa na qual as lixeiras podem estar cheias pode ser analisada com base em dados históricos e o transbordamento pode ser previsto antes que ele ocorra. Com a seleção otimizada das latas de lixo a serem recolhidas é esperado a redução de custos e a melhoria da eficiência de coleta, dependendo dos requisitos econômicos pré-definidos.

Monitoramento da qualidade do ar em áreas com aglomeração de pessoas, parques ou trilhas é outro serviço que o uso de IoT no ambiente urbano pode oferecer. Dispositivos podem se conectar a uma infraestrutura específica (p. e., um shopping) para saber sobre a qualidade do ar de certa região e também podem ser notificados sobre isso. Para a realização desse serviço, sensores devem ser implantados na cidade e os dados dos sensores disponibilizados aos cidadãos. Juntamente com esse serviço, um IoT urbano pode oferecer um serviço para monitorar o consumo de energia de toda uma cidade, tornando possível as autoridades e aos cidadãos obter uma visão clara e detalhada da quantidade de energia requerida pelos diferentes serviços, como por exemplo, iluminação pública, transporte, semáforos, controle de câmeras, aquecimento e resfriamento de construções públicas. As principais fontes de energia poderiam ser identificadas e seu uso poderia ser otimizado de forma a minimizar o consumo energético.

Dentro deste contexto, Rede Inteligente é um sistema de rede elétrica que utiliza TICs para coletar dados, tais como informação sobre fornecedores e consumidores e atua gerando valores à concessionária da rede. É constituída por uma rede elétrica, uma rede de comunicações e hardware ou software para monitoração e controle da rede. Isso envolve colocar sensores inteligentes e medidores inteligentes na produção, transmissão e no sistema de transmissão de energia, além dos pontos de acesso dos consumidores (NUAIMI et al., 2015). Isso para que dados em tempo real sejam obtidos sobre a produção de energia, consumo e possíveis falhas na rede (NUAIMI et al., 2015). Lembrando que um sistema medidor inteligente (*Smart Meters*) consiste de contadores inteligentes, infraestrutura de comunicação e dispositivos de controle. Os contadores inteligentes podem calcular o uso de energia elétrica e fornecer informações para a empresa regulamentar a energia, monitorar e controlar dispositivos. Em resumo, a rede fornece energia, minimiza custos e fornece informações instantâneas. Um exemplo comum é a rede elétrica digital que recolhe e distribui informações. Ela fornece eletricidade do fornecedor aos consumidores.

O ruído é visto como uma forma de poluição acústica. Neste sentido, em algumas cidades já existem leis específicas para reduzir os níveis de ruído nas áreas centrais em certos horários do dia. O uso de IoT no ambiente urbano (IoT urbano) pode oferecer um

serviço de monitoramento de ruído produzido em um dado horário nos lugares que adotam o serviço. Além de construir um mapa tempo-espaço de poluição sonora da área, o serviço poderia ser usado para aumentar a segurança pública. Isso poderia ser conseguido por meio de algoritmos de detecção de som que podem reconhecer o ruído de vidros quebrando ou brigas, por exemplo. A instalação de detectores de som ou microfones ambientais (sensores) seria interessante, embora existam algumas preocupações em relação à privacidade.

O monitoramento do congestionamento de tráfego na cidade é outro serviço que pode se tornar possível com o uso de IoT no ambiente urbano. Apesar dos sistemas de monitoração de tráfego baseados em câmeras estarem disponíveis e implementados em muitas cidades, a monitoração de tráfego poderia ser realizada em conjunto com a capacidade de sensoriamento e GPS instaladas nos veículos modernos e também a adoção de uma combinação de sensores acústicos e de qualidade do ar implantados ao longo de uma dada rua. Essa informação de congestionamento de tráfego é de grande importância para os cidadãos e para as autoridades da cidade. O tráfego seria mais disciplinado e quando necessário polícia ou bombeiros poderiam ser enviados aos locais. Rotas de trânsito poderiam ser planejadas de acordo com a qualidade do ar, os níveis de ruído e a quantidade de trânsito. Por outro lado, Sistemas de Transporte Inteligentes (STIs) aproveitam as novas tecnologias para melhorar a segurança, eficiência e conveniência do transporte terrestre, tanto para as pessoas quanto para os veículos de transporte (DIMITRAKOPOULOS; DEMESTICHAS, 2010). STIs contribuem para a exploração racional das infraestruturas existentes, sem recorrer à criação de novas instalações. STIs utilizam técnicas modernas de tecnologia de comunicação e meios de comunicação em áreas urbanas para o sistema de táxi, de trânsito rápido em massa, de trânsito ferroviário, pedágio eletrônico, sistema de gestão de informação rodoviária, sistema de otimização de sinal de trânsito, sistema de comunicações eletrônicas e sistemas de navegação em automóveis para enfrentar muitos desafios em vários meios de transporte.

O serviço de estacionamento inteligente é outro serviço possível com o uso de IoT no ambiente urbano. Ele é baseado em sensores implantados em ruas e displays inteligentes que direcionam os motoristas para o melhor caminho para estacionar seu carro em determinada rua. Os benefícios são múltiplos como a diminuição do tempo de busca de uma vaga de estacionamento, o que implica em menos emissão de $C0_2$, menor congestionamento de tráfego e satisfação dos cidadãos.

Uma Cidade Inteligente utiliza uma variedade de técnicas e sistemas que contribuem para reduzir o uso da água. Assim, faz-se necessário ter sistemas inteligentes para manter essa riqueza natural, os quais estão preocupados em: (i) Monitoramento e controle da água ambiental como a chuva natural, as águas superficiais, as águas subterrâneas, as águas residuais e as águas da agricultura; (ii) Análise e resposta aos dados para melhorar a eficiência do uso, o que requer cooperação com todos os setores interessados da sociedade;

(iii) Assegurar a segurança e saúde da rede certificando-se que a manutenção contínua é realizada; (iv) Controlar a poluição e fortalecer a capacidade de resposta em caso de emergência; (v) Usar medidores inteligentes que prevejam o consumo de água da população; (vi) Projetar espaços inteligentes que ajudem a reduzir a evaporação, e; (vii) Usar plantas e árvores que necessitem de pouca água.

Em relação a casas inteligentes, o objetivo é satisfazer o proprietário ou o ocupante. A casa inteligente obedece aos desejos do proprietário, em termos de proteção e conforto. Uma solução que pode ser fornecida por um edifício inteligente é o controle de iluminação. Um sistema de iluminação inteligente fornece iluminação em todos os lugares de modo que o ocupante nunca precise entrar em um quarto escuro. Controles de energia e temperatura proporcionam resfriamento ou aquecimento da casa. Segurança e proteção são fornecidos pela temperatura e sensores de movimento, que também podem desligar as luzes e fechar as portas quando você sair, além de disparar um alarme se um intruso aparecer. Entrada e saída são controlados por códigos de passagem inseridos através de um teclado. Algumas tecnologias que suportam a comunicação da casa inteligente com o proprietário são o computador e redes de Internet móveis.

Para cuidados médicos em Cidades Inteligentes, a cooperação entre os hospitais locais e redes privadas para o intercâmbio das informações é necessária. Isso para aumentar a eficiência para o tratamento dos pacientes e a vinculação com farmácias para fornecer medicamentos de forma rápida e facilmente. Acesso a redes sem fio para serviços de emergência pode ajudar as emergências médicas na transmissão de informações vitais dos pacientes. Diagnóstico automatizado e cuidados de saúde podem ser fornecidos para o paciente em uma situação perigosa. Dispositivos sensores podem ser colocados na roupa do paciente ou na pele e as informações enviadas para o hospital. Os dispositivos podem monitorar a temperatura, taxa de respiração, entre outros, em tempo real.

Um sistema de alimentação inteligente consiste em um sistema de rastreamento que monitora o abastecimento de alimentos, produção, processamento, transporte e controle de riscos. Outro elemento importante em um sistema de alimentação inteligente é assegurar a segurança dos alimentos. Um sistema de emergência pode fornecer alerta precoce de problemas de segurança alimentar.

Por fim, os serviços de emergência podem se beneficiar do uso de IoT em um contexto urbano de forma que drones poderiam supervisionar um atendimento de emergência, objetos inteligentes poderiam ser espalhados na cidade para detectar situações de emergência ou realizar o próprio acionamento das unidades móveis, ou as unidades móveis poderiam ser equipadas com sensores para captar dados do ambiente.

2.3.1 Serviços de Emergência

Os serviços de emergência ou serviços de resposta à emergência (*Emergency response*) abordam a resposta a um desastre natural ou artificial e envolve o controle da situação de desastre (CHITUMALLA et al., 2008). O processo é crítico em termos de tempo porque vidas dependem da precisão e rapidez com que o atendente do serviço de emergência toma decisões.

Durante uma resposta à emergência, os Centros de Operações de Emergência (*Emergency Operations Centers - EOCs*) enfrentam problemas diferentes, dependendo da categoria em que a situação de emergência se enquadra como por exemplo, incêndios, acidentes de trânsito, liberação de produtos químicos perigosos (CHITUMALLA et al., 2008). Um dos problemas é em relação à quais unidades móveis (veículos propriamente equipados e com profissionais qualificados) devem ser enviadas ao local da emergência em questão, ou seja, o problema de acionamento das unidades móveis mais adequadas para atuar em uma situação em particular.

a) Situações de Emergência

As situações possíveis de emergência em uma cidade são inúmeras e podem variar significativamente quanto às necessidades de urgência, de intensidade e de diversidade de ação, como ilustram os seguintes exemplos:

- Uma situação de assalto em um estabelecimento comercial requer mais urgência de ação por parte da polícia que uma situação de vandalismo contra edificações.
- Uma situação de incêndio de grandes proporções em um edifício requer uma ação de maior intensidade, isto é, com a participação de um número maior de bombeiros e equipamentos contra incêndio do que em uma situação de princípio de afogamento de uma pessoa em um rio.
- Uma situação de acidente de trânsito, pode requerer uma ação conjunta entre polícia, bombeiros e hospitais, isto é, essa ação apresenta uma diversidade de três tipos de serviços, e não de apenas um tipo, como nos exemplos anteriores (somente polícia ou somente bombeiros).

Invariavelmente, uma situação de emergência requer o deslocamento de uma ou mais unidades moveis, isto é, veículos apropriadamente equipados e com pessoal técnico especializado. Assim, uma situação de emergência necessita de um **serviço** ou mais para atendê-la, possui uma certa **urgência** e requer uma **intensidade**, ou seja, um determinado número de unidades móveis. Cada serviço (p. e., polícia, bombeiro, hospital) pode ter seu

próprio grau de urgência que pode ser baixo, médio, alto e muito alto, bem como uma intensidade diferente.

Na prática, observa-se que, independentemente do grau de urgência, espera-se que o provimento dos serviços ocorra sempre com a maior brevidade possível; qualquer atraso no provimento é justificável somente se houver outra situação com maior grau de urgência sendo atendida, de forma que as unidades moveis sejam, momentaneamente, insuficientes. O provimento de serviços em uma situação de emergência ocorre em, ao menos, duas fases, a saber:

a: Fase de acionamento dos serviços: Nesta fase, os correspondentes serviços devem ser comunicados da situação de maneira rápida e confiável. Quando comunicado, um serviço determina a urgência e a intensidade da sua ação específica e, com isso, decide quantas e quais unidades móveis serão enviadas ao local da emergência.

b: Fase de execução dos serviços: Nesta fase, as unidades móveis deslocam-se até o local da emergência e, quando chegam, executam a ação propriamente dita. Nessa execução, as unidades móveis podem precisar de informações atualizadas sobre a situação de emergência para fazer ajustes nas suas ações. Ainda nesta fase, é possível que novos acionamentos de serviço ocorram como consequência de revisão das dimensões diversidade e intensidade da ação. Ou seja, a classificação da situação de emergência pode mudar dinamicamente.

Portanto, o problema de provimento de serviços em situações de emergência é complexo e, para efeito de análise e pesquisa, pode ser dividido em problemas menores, porém fundamentais, da seguinte forma:

1. Definição da estratégia inicial de ação: Classificar a situação de emergência, determinando os tipos de serviços necessários, bem como a urgência e a intensidade de cada um.

2. Acionamento de unidades móveis: Selecionar e comunicar um conjunto de unidades móveis que satisfaça a necessidade de serviços (em diversidade e intensidade), de forma que os serviços sejam executados pelas unidades mais adequadas no momento, considerando as suas características particulares, incluindo disponibilidade de equipamentos e técnicos, bem como a sua distância relativa do local da emergência para que o serviço seja executado com a maior brevidade possível.

3. Apoio na execução: Supervisionar a execução da ação e orientar as unidades móveis na execução das suas ações específicas, fornecendo informações atualizadas sobre a situação de emergência.

4. Avaliação permanente da estratégia de ação: Avaliar permanentemente a eficácia da estratégia de ação em execução e, se necessário, readequar a estratégia através de alterações na sua classificação, isto é, nos tipos de serviços e correspondentes urgência e intensidade.

Uma Cidade Inteligente deve dispor de mecanismos que auxiliem na solução de cada um desses problemas, preferencialmente, com o mínimo de intervenção humana, tanto para se obter maior eficácia no provimento dos serviços, como para o uso mais eficiente dos recursos, em especial, as unidades móveis.

2.4 Cidades Inteligentes Espalhadas pelo Mundo

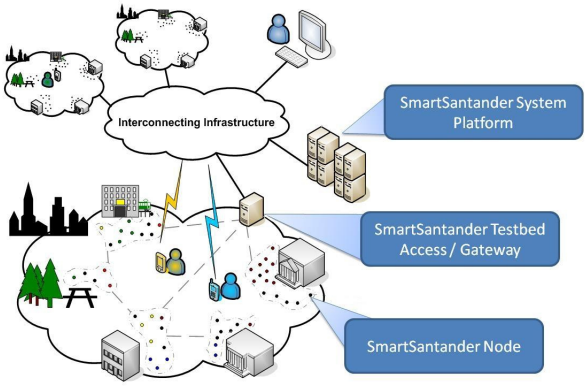
Muitas cidades espalhadas pelo mundo têm lançado iniciativas para tornarem-se Cidades Inteligentes, cada uma com suas diferenças. SmartSantander (Espanha), Pádua (Itália), Amsterdam (Holanda), Chicago (Estados Unidos) e São Paulo (Brasil) são apenas alguns exemplos de cidades que estão se tornando Cidades Inteligentes. Não existe uma fórmula para uma cidade tornar-se uma Cidade Inteligente. Em vez disso, um conjunto de transformações têm que ser realizadas para levar em conta diversos aspectos como, por exemplo, a melhora da mobilidade urbana, a melhora da segurança, o aumento da participação do cidadão no desenvolvimento da cidade, a criação de novos modelos econômicos que estimulem novas oportunidades de negócios, a melhoria do fornecimento e qualidade dos serviços médicos e educação, entre outros (PELLICER et al., 2013).

2.4.1 Santander (Espanha)

A cidade de Santander, ao norte da Espanha, é conhecida como Cidade Inteligente. Na cidade, um *testbed* foi desenvolvido e tem sido utilizado como um facilitador de testes experimentais para pesquisa e experimentação de arquiteturas, tecnologias, serviços e aplicações de Internet das Coisas para Cidades Inteligentes (SANCHEZ et al., 2014). O *testbed* contém mais de 15.000 sensores (anexados a 1.112 nós sensores) instalados na cidade. Alguns sensores fixos são colocados na infraestrutura das ruas, como em lâmpadas nas ruas, construções, postes, estacionamentos. Outros sensores são dinâmicos e encontram-se na rede de transporte público, como por exemplo, ônibus, taxis e carros de polícia. Os sensores implantados oferecem informações em tempo real em relação a diferentes parâmetros ambientais (p. e., luz, temperatura, ruído, emissão de CO_2), bem como informações sobre ocupação de vagas de estacionamento em algumas áreas centrais da cidade. O foco do projeto SmartSantander foi conectar a cidade de Santander através da implantação de sensores (SANCHEZ et al., 2014). O *testbed* SmartSantander oferece uma infraestrutura básica para coleta de dados, mas não prove qualquer capacidade de armazenamento, processamento e análise dos dados gerados (Figura 4). Uma vez que o

testbed não mantém dados históricos, as aplicações têm que gerenciar os dados coletados e realizar o processamento e análise com seus próprios recursos. Além disso, construir uma plataforma no topo do *testbed* SmartSantander para aplicações de Cidades Inteligentes é um desafio.

Figura 4 – Infraestrutura do *testbed* SmartSantander.



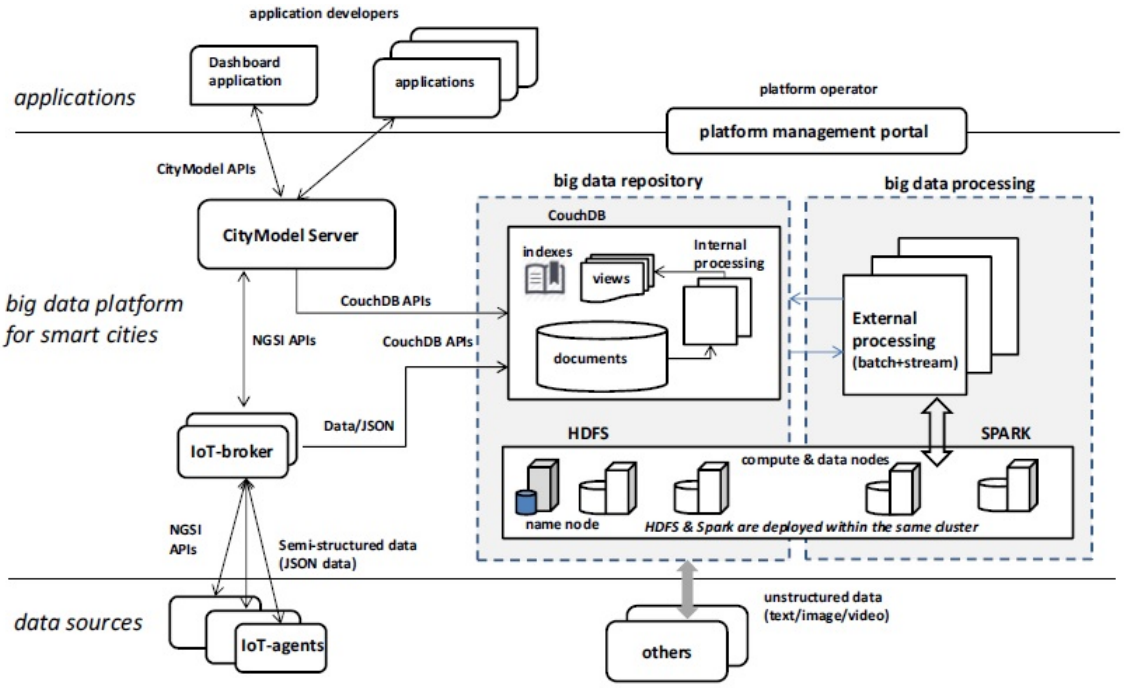
Fonte: http://www.smartsantander.eu/

Em (CHENG et al., 2015), é apresentada uma plataforma de Big Data para uma Cidade Inteligente chamada CIDAP, a qual trata dados históricos, dados próximos ao tempo real (*near-time data*) e também dados em tempo real. A plataforma foi integrada ao *testbed* SmartSantander e possui 4 módulos conforme mostra a Figura 5: (1) *IoT-agents* - Agentes de Internet das Coisas para coleta de dados; (2) *Big Data repository* - Repositório de Big Data para armazenamento de dados; (3) *Big Data Processing* - Processamento de Big Data para processamento de dados e análise, e; (4) *CityModel Server* - Servidor CityModel para comunicação com aplicações externas.

Dados de diferentes formatos são coletados através do IoT *broker* de múltiplas fontes, as quais são representadas pelos IoT-agents, e encaminhados para o Repositório de Big Data para armazenamento. Depois disso, os dados são processados e agregados por um conjunto de tarefas de processamento novas ou pré-definidas. Essas tarefas de processamento simples podem ser realizadas pelo Repositório de Big Data, o qual transforma dados em novos formatos; cria novas tabelas/visões estruturadas para indexar os dados. As tarefas de processamento intensivo ou complexas (p.e., agregação, mineração de dados) são separadas do Repositório de Big Data e são executadas pelo módulo de Processamento de Big Data (baseado em Spark - http://spark.apache.org/). Se não for necessário processamento de dados intensivo ou análise, o módulo de Processamento de Big Data é opcional. O CityModel Server é projetado para consultas e subscrições de aplicações externas baseadas em APIs pré-definidas CityModel, que são responsáveis por buscar os resultados gerados do Repositório de Big Data ou pelo encaminhamento de mensagens diretamente das fontes

de dados do *testbed*. Um portal web de gerência da plataforma é oferecido para monitorar o status de toda a plataforma. Um esquema da plataforma CiDAP é ilustrado na Figura 5.

Figura 5 – Plataforma CiDAP construída sobre o *testbed* SmartSantander.



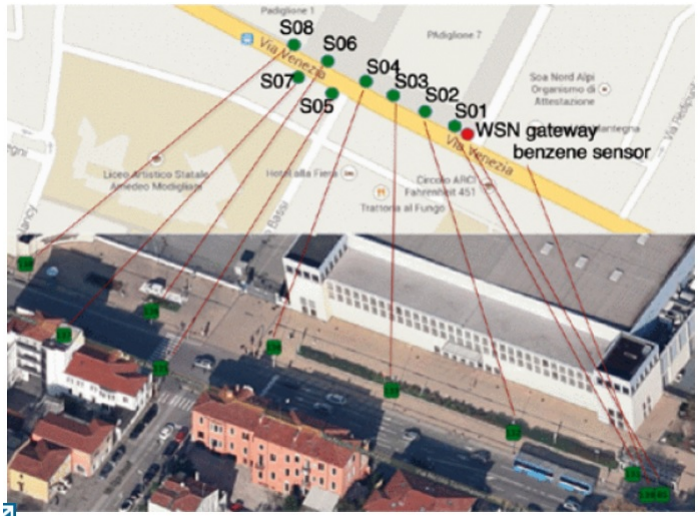
Fonte: (CHENG et al., 2015)

O projeto executou durante três meses, sendo que 50GB de dados foram coletados de 1.112 nodos sensores e existiam 9 tipos de nós sensores reportando seus valores de atualização para a plataforma CiDAP em vários intervalos de tempo. O intervalo de tempo foi pré-configurado para cada tipo de nó sensor respeitando sua economia de energia e tempo de vida da bateria. Cada mensagem de atualização possuía um tamanho de 536 bytes em média. No total 20 milhões de documentos únicos foram salvos no banco de dados CouchDB, resultando em 22GB de dados. O restante do espaço em disco foi utilizado para indexes e visões gerando um overhead adicional. A plataforma apresentou bons resultados em relação à taxa de transferência para tratar as consultas. Essa foi apenas uma plataforma que fez uso do *testbed* SmartSantander; entretanto, várias outras plataformas podem ser desenvolvidas com base na infraestrutura existente, bem como várias aplicações, como por exemplo (SANCHEZ et al., 2014): monitoração ambiental (p. e., monitoração da qualidade do ar, monitoração de níveis de ruído e monitoração dos níveis de luminosidade), gestão de estacionamento e orientação do condutor, irrigação de precisão em parques e jardins, realidade aumentada e sensoriamento participativo (ou seja , telefones móveis são usados como sensores).

2.4.2 Padova (Itália)

Padova é outra cidade que está se tornando uma Cidade Inteligente. O projeto Padova Smart City (PSC) é uma implementação prática de um IoT urbano realizado na cidade de Padova na Itália (CENEDESE et al., 2014). PSC é o resultado da colaboração entre entidades públicas e privadas, como o município de Pádua que foi o patrocinador do projeto; a Universidade de Padova, a qual forneceu a análise de viabilidade do projeto e os dados pós-processamento; e, um setor específico da Universidade de Padova especializado no desenvolvimento de soluções IoT inovadoras, o qual desenvolveu o software de controle dos nós IoT empregados no projeto. A aplicação principal do Padova Smart City consistiu de um sistema de monitorização da iluminação pública por meio de nós sem fios, equipados com diferentes tipos de sensores, colocados em postes de iluminação pública e ligados à Internet por meio de um *gateway* para recepção de dados e envio através da Internet. Além das medições de intensidade luminosa em cada poste, que podem ser utilizadas para verificar o funcionamento correto do sistema de iluminação pública, o sistema permitiu obter outros parâmetros ambientais como o nível de CO_2, temperatura e umidade do ar que fornecem dados sobre as condições meteorológicas, vibrações, ruído, entre outros. A Figura 6 ilustra a implantação de oito nós sensores implantados em uma avenida da cidade.

Figura 6 – Demonstração de nós sensores implantados em uma avenida com tráfego intenso.



Fonte: (CENEDESE et al., 2014)

Alguns dados coletados pelo sistema PSC mostraram como o processamento simples de dados pode fornecer informações interessantes relacionadas ao sistema de iluminação pública e aos níveis de poluição do ar.

2.4.3 Amsterdam (Holanda)

Amsterdam, na Holanda, possui um projeto de Cidade Inteligente nomeado de *Amsterdam Smart City (ASC)* (Ger Baron Amsterdam Innovation Motor, 2016). O projeto é uma parceria entre os habitantes de Amsterdam, as empresas e o governo para demonstrar como a energia pode ser economizada. Dentre alguns exemplos de subprojetos cita-se (Ger Baron Amsterdam Innovation Motor, 2016): Vida sustentável – Geuzenveld, West Orange, "Onze energy" e Gerência de energia; Espaço público sustentável – Climate Street e Smart Schools; Mobilidade Sustentável – Ultra fast charging.

Em Geuzenveld, vizinha a Amsterdam, medidores inteligentes foram implantados em várias casas para aumentar a conscientização das pessoas sobre seu consumo de energia com o objetivo de alterar o comportamento do consumidor. De forma semelhante em West Orange displays foram conectados a medidores inteligentes em várias casas para controlar o aquecimento central (gás ou energia elétrica). Com isso dispositivos domésticos seriam melhor utilizados ou substituídos e o aquecimento poderia ser diminuído ou desligado levando, como consequência, a economia de energia e/ou gás. "Onze Energy" (Our Energy) é um conceito de financiamento coletivo de sete moinhos de vento com a ambição de chegar a 20 % da população de Amsterdam tornando-se membro desta cooperação. Gerência de energia consiste em testar tomadas inteligentes para que os consumidores tenham um melhor insight do uso de energia e possam pensar em meios de economizá-la, bem como propor sistemas de gerência de energia mais efetivos.

Climate Street é outro subprograma que possui algumas iniciativas como criar logísticas para diminuir o tráfego nas ruas da cidade, coletar o lixo com veículos elétricos, realizar o ofuscamento da luz pública, utilizar medidores e tomadas inteligentes. Smart Schools trata, dentre outros objetivos, de conscientizar crianças sobre o uso eficiente de energia. A cidade de Amsterdam possui veículos elétricos e o subprograma Carregamento ultra rápido (*Ultra fast charging*) trata de carregar as baterias desses veículos em mais ou menos 20 minutos.

2.4.4 Chicago (Estados Unidos)

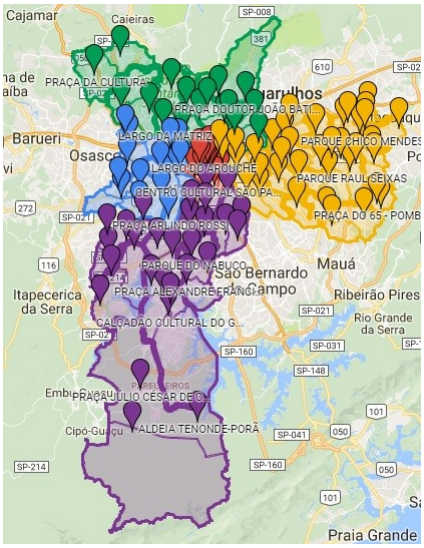
Chicago, nos Estados Unidos, é outra cidade em direção a tornar-se uma Cidade Inteligente. A cidade possui um projeto nominado de *Array of Things*, que envolve a instalação de cerca de 500 nós sensores nas ruas da cidade [2]. Os sensores coletarão dados sobre tráfego, qualidade do ar, níveis sonoros, temperatura e níveis de água em ruas e calhas. As informações dos sensores ajudarão no planejamento urbano e estarão disponíveis para os cidadãos analisarem. A iniciativa da Cidade Inteligente de Chicago também tem o apoio de muitas empresas locais, do estado de Illinois e do governo federal.

[2] http://www.ioti.com/smart-cities/why-chicago-smart-city-king

2.4.5 São Paulo (Brasil)

Na cidade de São Paulo, uma primeira iniciativa em direção a implementação do Conceito de Cidades Inteligentes é o programa WiFi LivreSP [3], o qual tem como objetivo levar Internet gratuita e de qualidade aos cidadão da cidade. A velocidade mínima da Internet oferecida é de 512 kbps por usuário, tendo uso irrestrito pelos cidadãos que tenham um dispositivo compatível com o protocolo WiFi (p. e., laptops, celulares, tablets). O processo de implantação do Programa WiFi LivreSP durou um ano e quatro meses, com inauguração em janeiro de 2014. O número de localidades públicas atendidas atualmente é de 120, que são distribuídas em cinco regiões da cidade (Figura 7): Centro, Norte, Sul, Leste e Oeste. O programa encontra-se em expansão, devido à demanda da comunidade, para novas localidades.

Figura 7 – Localidades em que o programa Wifi LivreSP se faz presente: Norte (Verde), Sul (Roxo), Leste (Laranja), Oeste (Azul), Centro (Vermelho).



Fonte: https://wifilivre.sp.gov.br/

Embora São Paulo esteja em direção da implementação do conceito de Cidades Inteligentes, essa iniciativa é melhor classificada como Cidade Digital. Cidade Digital diz respeito a obter e organizar a informação da cidade em uma forma digital provendo uma área digital na qual moradores e visitantes conseguem interagir entre eles (REZENDE et al., 2014). Para muitos, o conceito de Cidades Digitais implica no oferecimento de Internet para todos os cidadãos.

[3] https://wifilivre.sp.gov.br/

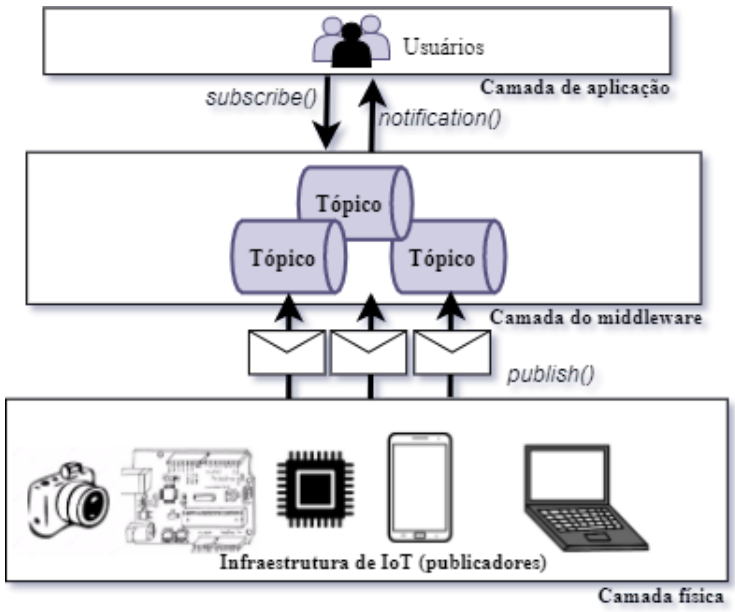
3 Serviço de Notificação de Eventos

Um serviço de notificação de eventos (*Event Notification Service*) ou serviço de eventos (*Event Service*) ou *middleware publish-subscribe* ou simplesmente *middleware* orientado a eventos é uma infraestrutura independente da aplicação que suporta a construção de sistemas baseados em eventos, no qual geradores de eventos notificam eventos (i. e., informação(ções) sobre algo) para a infraestrutura e consumidores de eventos se inscrevem com a infraestrutura para receber notificações relevantes (CARZANIGA; ROSENBLUM; WOLF, 2001). Em um serviço de eventos, os componentes (isto é, as partes que compõem o *middleware*), aplicações e todos os outros participantes interagem através de eventos (RAZZAQUE et al., 2016). Cada participante pode assumir o papel de um *publisher* ou publicador (i. e., produtor de eventos) ou de um *subscriber* ou assinante (i. e., consumidor de eventos) de informação, ou os dois ao mesmo tempo (BALDONI; VIRGILLITO, 2005). Assinantes possuem a capacidade de expressar seu interesse em um evento ou padrão de eventos e são subsequentemente notificados de qualquer evento, gerado por um publicador, que corresponde ao seu interesse registrado (EUGSTER et al., 2003). Publicadores, por outro lado, produzem informações na forma de eventos que são consumidos por assinantes emitindo subscrições (*subscriptions*) representando seu interesse em eventos específicos (BALDONI; VIRGILLITO, 2005). Em outros termos, produtores publicam informação em um gerente de eventos (*event manager*) e consumidores subscrevem-se para receber aquela informação. Com isso, os eventos são propagados dos publicadores para os assinantes (RAZZAQUE et al., 2016) de forma assíncrona para todos os assinantes que registraram interesse naquele dado evento (EUGSTER et al., 2003). Sempre que um evento é publicado, o serviço de eventos dispara o evento para um conjunto de assinantes que especificaram um filtro na subscrição que correspondeu (*matches*) ao evento (i. e., avaliada como verdadeiro quando aplicado ao evento). Essa interação entre publicadores e assinantes é mediada pelo serviço de notificação de eventos. Assim, a principal característica semântica de um serviço de eventos está na forma como os eventos fluem dos publicadores para os assinantes, ou seja, os assinantes não são diretamente endereçáveis pelo publicador, mas são endereçados indiretamente de acordo com o conteúdo dos eventos (BALDONI; VIRGILLITO, 2005). Esse anonimato garante que publicadores e assinantes troquem informações sem se conhecerem, o que permite que o serviço de eventos se expanda para um tamanho massivo na escala da Internet e da Internet das Coisas (*Internet of Things* – IoT). Consequentemente, em Cidades Inteligentes *middlewares publish-subscribe* podem ser utilizados.

3.1 Conceito

Um serviço de notificação de eventos é definido como o *middleware* que implementa um modelo de eventos, como consequência oferecendo comunicação baseada em eventos para um sistema de eventos (MEIER; CAHILL, 2002). Um modelo de eventos consiste de um conjunto de regras descrevendo um modelo de comunicação que é baseado em eventos (MEIER; CAHILL, 2002). Um sistema baseado em eventos ou simplesmente um sistema de eventos é uma aplicação que utiliza um *middleware* baseado em eventos para permitir que componentes que compreendem a aplicação interajam por meio de notificações de eventos. Os sistemas de eventos que exploram um *middleware* baseado em eventos são organizados como uma coleção de componentes autônomos, os clientes, os quais interagem por meio de eventos e por subscrições para as classes de eventos para os quais eles estão interessados (CUGOLA; JACOBSEN et al., 2002). Clientes, neste contexto, se referem aos publicadores e assinantes de eventos. Em resumo, um sistema de eventos utilizando um *middleware* baseado em eventos consiste de um grande número de componentes de aplicação, ou entidades, que produzem e consomem eventos (MEIER; CAHILL, 2002). A Figura 8 apresenta um *middleware* baseado em eventos no cenário de IoT, no qual tem-se um conjunto de publicadores e assinantes. Os publicadores são os objetos inteligentes e os assinantes são os usuários que se encontram na camada de aplicação.

Figura 8 – Modelo de projeto para um *middleware* baseado em eventos em IoT.

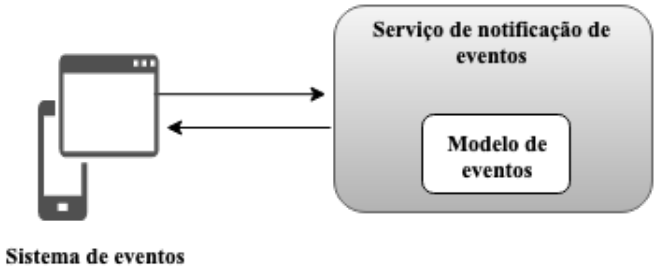


Fonte: (RAZZAQUE et al., 2016)

A principal vantagem desse tipo de serviço de eventos é o forte desacoplamento de espaço, tempo e sincronismo entre os publicadores e os assinantes dos eventos (EUGSTER

et al., 2003). Desacoplamento de espaço porque publicadores não precisam saber quem são os assinantes e estes, por sua vez, não necessitam tomar conhecimento de quem são os publicadores. Desacoplamento de tempo porque as partes interagindo podem não participar ativamente na interação ao mesmo tempo. Além de que, no desacoplamento de sincronismo publicadores e assinantes podem realizar atividades simultâneas enquanto eventos são produzidos e consumidos. Como consequência, publicadores e assinantes são independentes e a arquitetura global do *middleware* se torna mais flexível, porém mais complexa (MAGNONI, 2015). A Figura 9 apresenta uma visão geral de um *middleware* orientado a eventos, sendo o *middleware* o serviço de eventos, o sistema de eventos a aplicação que utiliza o *middleware* e o modelo de eventos as regras descrevendo como se dá a comunicação entre os componentes do *middleware*.

Figura 9 – Visão geral de um *middleware* baseado em eventos.



Fonte: Autoria própria

O serviço de notificação de eventos deve implementar dois serviços que são (Carzaniga et al., 2001):

i A seleção do evento (*notification selection*), a qual consiste em determinar quais eventos ou notificações corresponde às subscrições, também conhecida como *matching*, e;

ii A entrega do evento ou notificação (*notification delivery*), isto é, encaminhar os eventos correspondentes dos publicadores aos assinantes. A fase de *matching* é necessária para não sobrecarregar a rede de mensagens.

Alguns exemplos de sistemas de eventos onde este tipo de *middleware* pode ser utilizado são a negociação de ações, anúncios de compra e venda, boletins meteorológicos, boletins de condições de tráfego, notificações de eventos do Facebook, entre outros.

3.1.1 Evento

A informação trocada entre publicadores e assinantes é denotada pelo termo evento ou notificação de eventos ou simplesmente notificação. Um evento normalmente é estruturado como um conjunto de pares atributo-valor, no qual cada atributo possui um tipo (p. e., inteiro, real, string), um nome e um valor (BALDONI; VIRGILLITO, 2005). Por exemplo, um evento relacionado a notícias teria a forma: string evento = "notícias".

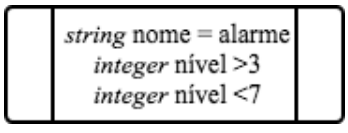
Eventos são encontrados em duas formas (EUGSTER et al., 2003): mensagens ou invocações. No primeiro caso, as mensagens são entregues a um assinante através de uma única operação genérica, normalmente executada no publicador. Isto significa que o *matching* acontece no publicador. No segundo caso, os eventos disparam a execução de operações específicas do assinante, ou seja, invocações são realizadas no assinante quando o evento é recebido. Neste caso, o *matching* acontece no assinante.

Cabe salientar que normalmente serviços de eventos não oferecem persistência de eventos, uma vez que as mensagens são enviadas diretamente pelo publicador para todos os assinantes. A menos que o produtor mantenha uma cópia de cada mensagem, um assinante defeituoso pode não conseguir receber mensagens perdidas durante sua recuperação (EUGSTER et al., 2003).

3.1.2 Filtros de Eventos

Filtros de eventos ou simplesmente filtros selecionam notificações de eventos por especificar um conjunto de atributos e restrições sobre os valores dos atributos do evento (CARZANIGA; ROSENBLUM; WOLF, 2001). Essa especificação sobre os valores dos atributos é realizada quando os assinantes realizam as suas inscrições para receber os eventos. Cada restrição de atributo é uma tupla especificando um tipo, um nome, um operador relacional binário e um valor para um atributo. A Figura 10 ilustra um filtro de evento, no qual o consumidor especificou que um alarme deve ser gerado quando o nível se encontrar entre 3 e 7 e o nome do evento deve ser alarme.

Figura 10 – Exemplo de um filtro de evento.



Fonte: (CARZANIGA; ROSENBLUM; WOLF, 2001)

Quando um filtro é utilizado em uma subscrição, múltiplas restrições são interpretadas como uma conjunção (CARZANIGA; ROSENBLUM; WOLF, 2001). Na Figura 10, isso significaria o filtro: ((nome = "alarme") AND (nível > 3) AND (nível < 7)). Logo,

um assinante recebendo um evento do tipo alarme realiza uma comparação entre o evento recebido e a subscrição realizada.

3.1.3 Matching em um Serviço de Eventos Publish-subscribe

O *matching* é um dos processos realizados pelo serviço de notificação de eventos e tem como função checar se um evento pertence a uma subscrição determinando assim, se o mesmo será disparado para o assinante ou não (BALDONI; VIRGILLITO, 2005). Desta forma, uma notificação ϵ corresponde a uma subscrição α se ela satisfaz todas as restrições declaradas nos atributos correspondentes. A tarefa é verificar se uma notificação combina com os filtros das subscrições. Observe a Tabela 2, na qual existe um evento e uma subscrição. No primeiro e no último caso, o evento será enviado aos assinantes porque o *matching* foi positivo ou então o evento ficará com o assinante se um evento do tipo invocação foi utilizado.

Tabela 2 – Exemplo de *matching*.

Notificação	***Matching***	**Subscrição**
string nome = “alarme”		
time data=02:40:03	Sim	string nome = “alarme”
string nome = “alarme”		
time data=02:40:03	Não	string nome = “alarme”; integer nível > 3
string nome = “alarme”		
integer nível = 10	Não	string nome = “alarme”; integer nível > 3; integer nível < 7
string nome = “alarme”		
integer nível = 5	Sim	string nome = “alarme”; integer nível > 3; integer nível < 7

Fonte: Adaptado de (CARZANIGA; ROSENBLUM; WOLF, 2001)

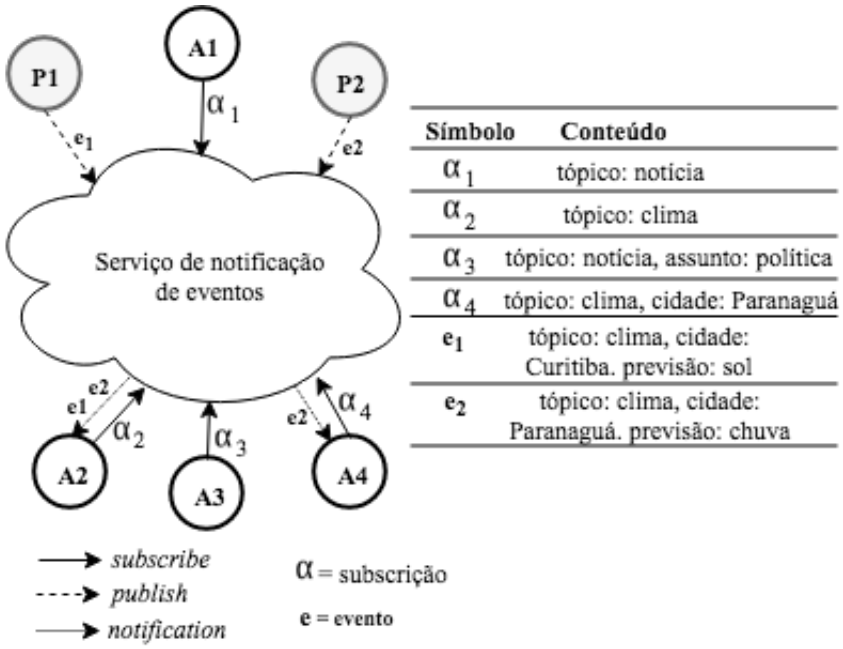
Para o encaminhamento de eventos, a fase de *matching* é necessária (BALDONI; VIRGILLITO, 2005).

3.2 Método Básico de Interação Publish-subscribe

Um serviço de notificação de eventos deve oferecer armazenamento e gerência de subscrições e entrega eficiente de eventos (EUGSTER et al., 2003). O serviço de eventos representa um mediador neutro entre publicadores e assinantes. O interesse no evento pelo assinante normalmente é realizado pela chamada de uma operação *subscribe()* no serviço de eventos. Os assinantes têm que se inscrever nos eventos nos quais eles estão interessados para recebê-los. A informação de subscrição é armazenada no serviço de

eventos e não é encaminhada aos publicadores. A operação *unsubscribe()* cancela uma subscrição. Assim, uma vez que os assinantes têm se subscrito em eventos, os quais estão interessados, eles recebem todos os eventos subsequentes até cancelarem sua subscrição (MEIER; CAHILL, 2002). Para gerar um evento, um publicador normalmente chama uma operação *publish()* no serviço de eventos. O serviço de eventos propaga o evento para todos os assinantes relevantes; ele pode assim ser visualizado como um *proxy* para os assinantes. Publicadores também podem anunciar os eventos que irão publicar por meio de uma operação *advertise()*. A informação oferecida pode ser importante para que o serviço de eventos ajuste-se ao fluxo de eventos e para que os assinantes fiquem sabendo quando um novo tipo de informação torna-se disponível. A Figura 11 ilustra de uma forma geral o método de interação básico em um serviço de eventos.

Figura 11 – Método básico de interação em um serviço de eventos.



Fonte: Autoria própria

Na Figura 11, A_1, A_2, A_3 e A_4 subscrevem-se com os filtros α_1, α_2, α_3 e α_4 por meio de *subscribe()*. P_1 e P_2 publicam os eventos e_1 e e_2 para o serviço de notificação de eventos por meio de *publish()* que notifica os assinantes A_2 e A_4 por meio de *notification()*. O assinante A_2 recebe os eventos e_1 e e_2 porque realizou uma subscrição para o evento clima. Isso acontece porque o *matching* foi positivo quando a subscrição de A_2 foi comparada com os eventos presentes no serviço de eventos. A_4 recebe somente o evento e_2 porque a subscrição foi mais específica, o clima da cidade de Paranaguá, e o *matching* da subscrição α_4 com o evento α_2 foi positivo, logo A_4 foi notificado.

3.3 Modelos de Subscrição

Assinantes normalmente estão interessados em eventos particulares ou padrões de eventos e não em todos os eventos. Os diferentes modos de especificar os eventos de interesse levam a vários modelos de subscrição, sendo os clássicos (EUGSTER et al., 2003): *publish-subscribe* baseado em tópicos, *publish-subscribe* baseado em conteúdo e *publish-subscribe* baseado em tipo.

3.3.1 Publish-Subscribe baseado em Tópicos

Participantes podem publicar eventos e subscrever-se em tópicos individuais, os quais são identificados por palavras-chave (EUGSTER et al., 2003). Notificações são agrupadas em tópicos, isto é, um assinante declara seu interesse em um tópico particular e recebe todos os eventos relacionados àquele tópico (MAGNONI, 2015). O assinante especifica um tópico e cada tópico corresponde a um canal lógico conectando cada possível publicador a todos os assinantes interessados. Tópicos são similares a noção de grupos no contexto de comunicação em grupo (EUGSTER et al., 2003). A inscrição para um tópico T pode ser visualizada como a entrada de um membro em um grupo T; e, publicar um evento em um tópico T traduz-se em difundir em *broadcasting* aquele evento entre os membros de T.

O *publish-subscribe* baseado em tópicos introduz uma abstração de programação com tópicos individuais mapeados a canais de comunicação distintos. O nome do tópico é normalmente especificado como um argumento de inicialização. Cada tópico é visualizado como um serviço de evento identificado por um nome único com uma interface oferecendo as operações *publish()* e *subscribe()*. Desta forma, os assinantes são conhecidos a priori e os tópicos também podem ser organizados na forma de uma hierarquia. A desvantagem deste modelo de subscrição é a expressividade limitada devido a seu modelo estático.

3.3.2 Publish-Subscribe baseado em Conteúdo ou Propriedade

O *publish-subscribe* baseado em conteúdo introduz um modelo de subscrição baseado no conteúdo dos eventos considerados (EUGSTER et al., 2003). Eventos são classificados de acordo com as suas propriedades. Desta forma, assinantes expressam seu interesse por especificar condições sobre o conteúdo das notificações que eles querem receber permitindo filtragem dependendo dos dados do evento (MAGNONI, 2015). Consumidores subscrevem-se para eventos selecionados por especificar filtros nas subscrições usando uma linguagem de subscrição. Os filtros definem restrições, geralmente na forma de pares de propriedades nome-valor e operadores de comparação básica (i. e., operadores relacionais) que identificam eventos válidos. As restrições podem ser combinadas logicamente (*and*, *or*, entre outros) para formar padrões de subscrição complexos. Padrões de subscrições são usados para

identificar os eventos de interesse de um assinante e propagar esses eventos de acordo com os padrões. Subscrições contêm expressões que permitem realizar o *matching* com o conteúdo do evento. Logo, na operação *subscribe()*, um argumento adicional representa uma restrição ou um padrão da subscrição (p. e., $restricao = (companhia == "XXX"$ $and\ preco < 120)$). Os padrões podem ser representados com o uso de *strings* e *template* de objetos. Quando uma subscrição é realizada utilizando um template de objetos, o assinante oferece um objeto t que indica que ele está interessado nos eventos do tipo t e cujos atributos correspondam aos atributos correspondentes de t, exceto aqueles que levam *null*.

O modelo *publish-subscribe* baseado em conteúdo é mais expressivo e mais geral do que o modelo baseado em tópicos. Entretanto, requer protocolos que terão maior *overhead* quando ocorre a publicação de eventos que o modelo baseado em tópicos, por causa das comparações que precisam ser realizadas. Neste modelo, os assinantes não podem ser determinados antes da publicação. Fazendo-se uma análise mais geral, o modelo baseado em tópicos pode ser visualizado como uma instância particular do modelo baseado em conteúdo em que o *matching* é realizado somente sobre o "*nome*" do evento.

3.3.3 Publish-Subscribe baseado em Tipo

O *publish-subscribe* baseado em tipo surgiu para substituir o modelo de subscrição *publish-subscribe* baseado em tópicos. Isso porque normalmente os tópicos reagrupam eventos que apresentam semelhanças em conteúdo e estrutura (EUGSTER et al., 2003). Desta forma, em uma subscrição baseada em tipo, a declaração de um tipo desejado é o principal atributo discriminante. O assinante especifica na subscrição o tipo que deseja receber e a correspondência de eventos é realizada com base no tipo especificado. Em resumo, os eventos são filtrados de acordo com seu tipo. Neste modelo, eventos são objetos pertencentes a um tipo específico, o qual encapsula atributos e métodos. Expressividade adicional pode ser adquirida aplicando filtros baseados em conteúdo no contexto de tipos para expressar restrições nos valores dos objetos.

Outros modelos de subscrição podem ser encontrados como o *publish-subscribe* baseado em conceito e *publish-subscribe* adaptável à localização (BALDONI; VIRGILLITO, 2005):

1. **Modelo baseado em conceito**: permite descrever um esquema de eventos em um nível de abstração mais alto por utilizar ontologias. Ontologia refere-se a um sistema particular de categorias levando em conta certa visão do mundo (GUARINO, 1995). Uma ontologia é um artefato de engenharia constituído de um vocabulário específico para descrever certa realidade, mais um conjunto de suposições explícitas sobre o significado pretendido das palavras desse vocabulário (GUARINO, 1995). O

objetivo de criar ontologias é para se ter um entendimento comum sobre um domínio específico a ser entendido por pessoas e computadores.

2. **Modelo adaptável à localização**: sistemas *publish-subscribe* utilizados em ambientes móveis requerem suporte a subscrições adaptáveis à localização. Por exemplo, um assinante móvel pode consultar o sistema para receber notificações quando ele está próximo de um local específico ou serviço. A implementação dessas subscrições requer ao sistema *publish-subscribe* a capacidade de monitorar a mobilidade dos clientes.

Autores definem diferentes modelos de subscrição. Para (EUGSTER et al., 2003), os modelos de subscrição considerados são os modelos baseados em tópico, conteúdo e tipo. Em (PIETZUCH; BACON, 2002), são dois os modelos de subscrição considerados, os modelos de subscrição baseados em tópico e conteúdo. Para (BALDONI; VIRGILLITO, 2005), os modelos de subscrição são os modelos baseados em tópico, conteúdo, tipo, conceito e adaptável a localização.

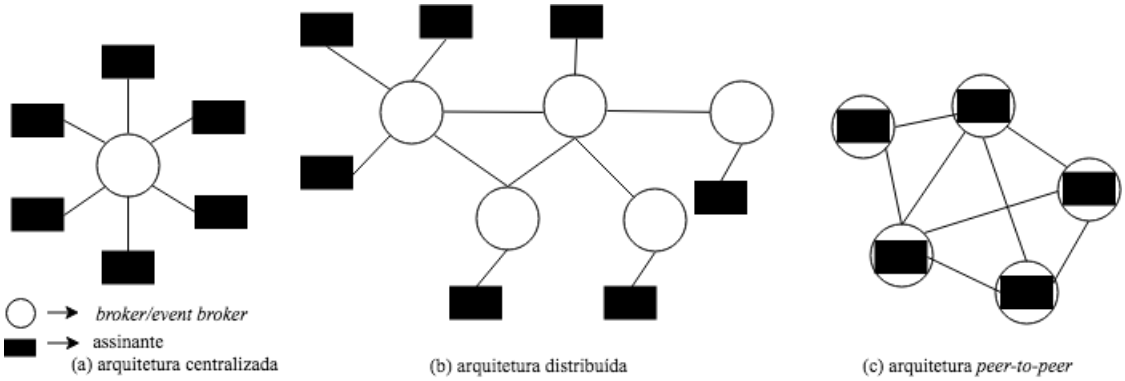
3.4 Arquitetura de um Serviço de Notificação de Eventos

A arquitetura de um serviço de notificação de eventos diz respeito à estruturação dos componentes de um sistema de eventos, a qual pode se centralizada, distribuída (*Overlay* de *brokers*) ou ainda *Peer-to-peer*, conforme pode ser visualizado na Figura 12 (CUGOLA; JACOBSEN et al., 2002). Na arquitetura centralizada, existe uma entidade central que é responsável por coletar subscrições e encaminhar eventos (CUGOLA; JACOBSEN et al., 2002). Essa entidade central é conhecida como *broker* ou *event broker*. O assincronismo é implementado por ter produtores enviando mensagens para essa entidade específica, o *broker*, que armazena e então encaminha as mensagens para os assinantes sob demanda (EUGSTER et al., 2003). Entretanto, isso depende do método de recebimento das mensagens. Se for *push*, o *broker* encaminha as mensagens para os assinantes, caso contrário, se for *pull*, o próprio assinante recupera as mensagens do *broker*. Nesta arquitetura, existe um único componente que age como despachante dos eventos introduzindo um único ponto de falhas, bem como diminui a escalabilidade quando o número de participantes ou a taxa de eventos cresce significativamente. Entretanto, permite fácil administração dos recursos disponíveis e simples implementação.

Na arquitetura distribuída não existe uma entidade central no sistema (EUGSTER et al., 2003), o que reduz a carga da rede e aumenta a escalabilidade. Nesta arquitetura, o serviço de notificação de eventos pode ser implementado como uma rede de servidores distribuídos, também conhecidos como servidores despachantes (*dispatching servers*), os quais colaboram na coleta de subscrições vinda de clientes e no roteamento de eventos

(MOTTOLA; CUGOLA; PICCO, 2008). O serviço de notificação de eventos acaba sendo constituído de um conjunto de nós distribuídos que coordenam entre eles mesmos a ordem de despachar eventos publicados para todos os (e possivelmente somente para os) assinantes interessados (BACON et al., 2000).

Figura 12 – Arquitetura do broker em um *middleware publish-subscribe*.



Fonte: Adaptação de (CUGOLA; JACOBSEN et al., 2002)

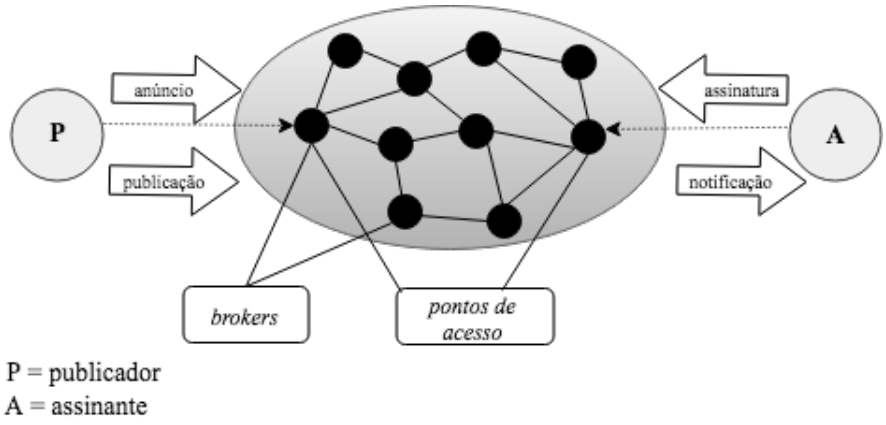
Em (BELLAVISTA; CORRADI; REALE, 2014), a arquitetura distribuída é classificada como *Overlay* de *brokers*. Os autores mencionam que o *middleware* é organizado como uma rede de pares (*peers*) de *brokers*, na qual a correspondência (*matching*) de eventos e o roteamento são realizados por algoritmos distribuídos sobre a redes de *brokers*, ou seja, pelos próprios *brokers*. As interações entre os participantes seguem o modelo cliente servidor. Este tipo de arquitetura torna possível a gerência de um grande número de participantes e eventos, porque a responsabilidade e a complexidade das funções de *matching* e as decisões de roteamento são divididas entre a rede *overlay* de *brokers*. A comunicação é assíncrona e anônima sem a necessidade de uma entidade intermediária, como é o caso da arquitetura centralizada. A topologia do(s) *broker(s)* de eventos e as estratégias adotadas para rotear subscrições e eventos modificam-se de sistema para sistema.

Outra arquitetura apresentada por (BELLAVISTA; CORRADI; REALE, 2014) é a arquitetura *Peer-to-peer*. Nesta arquitetura, o fluxo de eventos vai dos publicadores aos assinantes sem nós intermediários e todas as funções de *matching* e roteamento são realizadas pelos próprios participantes. O assincronismo é implementado pelo uso de primitivas de comunicação inteligentes que implementam mecanismos de encaminhamento e armazenamento em ambos processos produtores e consumidores (EUGSTER et al., 2003). Este tipo de arquitetura é adequada para a disseminação de eventos entre um número limitado de participante e com implantação em pequena escala geográfica devido a questões de latência e alto *throughput* de eventos.

De uma forma geral, o serviço de eventos ou *middleware* é implementado como

uma *Overlay* de *brokers* ou *brokers* que oferecem pontos de acesso aos clientes (Figura 13). Clientes utilizam os pontos de acesso para anunciar informações sobre eventos e consequentemente publicar múltiplas notificações do tipo previamente anunciado (CARZANIGA; ROSENBLUM; WOLF, 2001). Alguns serviços de eventos não disponibilizam pontos de acesso para o anúncio de eventos. Um anúncio expressa o interesse do cliente em publicar um tipo particular de notificação. Clientes também utilizam os pontos de acesso para se inscrever em notificações de interesse. O serviço então utiliza os pontos de acesso para notificar os clientes sobre a entrega de qualquer notificação de interesse. Essa rede de servidores pode ser visualizada como uma rede *overlay* sobre uma rede física, como a Internet, por exemplo.

Figura 13 – Serviço de notificação de eventos distribuído.



Fonte: (CARZANIGA; ROSENBLUM; WOLF, 2001)

Criar uma rede de servidores para prover um serviço distribuído qualquer dá origem a três pontos críticos em um projeto, os quais são apontados por (CARZANIGA; ROSENBLUM; WOLF, 2001):

1. Topologia de interconexão: Em qual configuração os servidores devem ser conectados?
2. Algoritmo de roteamento: Qual informação deve ser comunicada entre os servidores para permitir a entrega eficiente e correta de mensagens?
3. Estratégia de processamento: Onde na rede e de acordo com quais heurísticas, as mensagens devem ser processadas para otimizar o tráfego de mensagens?

Entretanto, para que esses três pontos sejam analisados o domínio do *middleware* deve ser analisado, ou seja, se o mesmo será utilizado em redes locais, redes sem fio ou então em Internet das Coisas.

3.5 Topologia de Brokers na Organização Distribuída

A topologia representa a organização lógica dos vários servidores (*brokers*) que compõe um serviço de eventos considerando uma arquitetura distribuída. Várias topologias podem ser criadas dependendo do ambiente em que os *brokers* estiverem, conforme pode ser visualizado na Figura 14. Alguns exemplos são a topologia hierárquica, a topologia *peer-to-peer* acíclica, a topologia *peer-to-peer* geral, além das topologias híbridas (CARZANIGA; ROSENBLUM; WOLF, 2001).

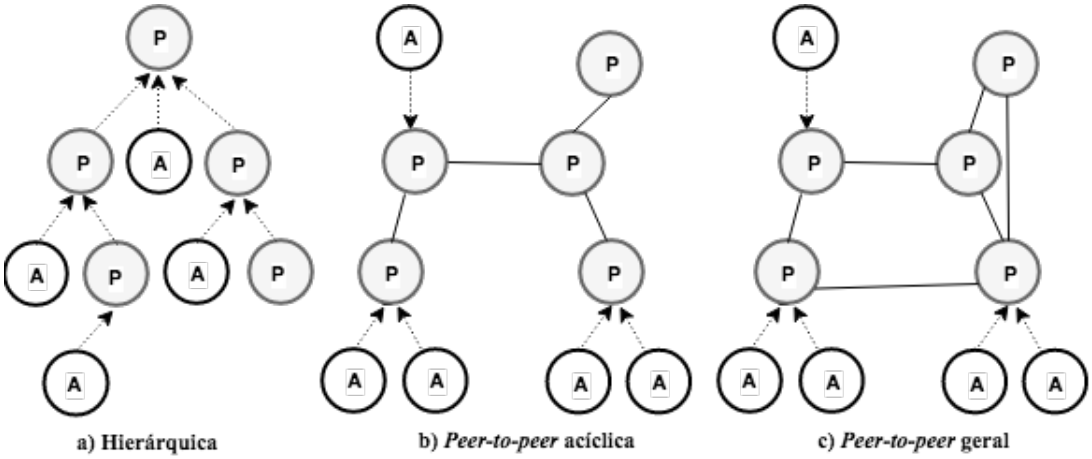
Na topologia hierárquica, pares de *brokers* conectados interagem em uma relação cliente/servidor assimétrica. Um *broker* pode ter qualquer número de conexões de entrada de outros *brokers* "clientes", mas apenas uma conexão de saída para o seu próprio *broker* "mestre", formando assim uma hierarquia. O *broker* que não tiver um servidor "mestre" é referido como raiz. Um grafo direcionado/orientado é utilizado para representar esta topologia, conforme ilustra a Figura 14 (a). Essa topologia é uma extensão da arquitetura centralizada, a única coisa é que a entidade central básica precisa ser modificada para propagar qualquer informação que recebe (p. e., subscrições) para o seu *broker* mestre. A principal desvantagem desta topologia é a sobrecarga dos *brokers* mais altos na hierarquia.

Na topologia *peer-to-peer* acíclica, *brokers* comunicam-se simetricamente como *peers*, adotando um protocolo que permite um fluxo bidirecional de subscrições, anúncios e notificações (Figura 14 (b)). Um grafo não direcionado acíclico é utilizado para representar esta topologia, ou seja, as conexões entre os *brokers* não contem ciclos. A principal desvantagem desta topologia e da topologia hierárquica está na falta de redundância. Se um *broker* falha, os *brokers* conectados a ele ficam isolados do resto da rede. Outra questão é em relação aos algoritmos de roteamento que devem tratar falhas se acontecerem.

Na topologia *peer-to-peer* geral é permitida a comunicação bidirecional entre dois *brokers*, mas a topologia pode formar um grafo não direcionado geral com a possibilidade de ter vários caminhos entre os servidores (Figura 14 (c)). A vantagem desta topologia sobre as outras é que ele oferece redundância devido aos vários caminhos entre os *brokers*; entretanto, algoritmos especiais de roteamento devem ser implementados para evitar ciclos na escolha dos melhores caminhos. Normalmente, terão um contador *time-to-live* e rotas serão estabelecidas de acordo com a árvore de extensão mínima (*minimal spanning tree*).

As topologias híbridas permitem utilizar diferentes topologias em diferentes níveis de granularidade da rede. A Figura 15 (a) ilustra uma topologia híbrida em que uma topologia *peer-to-peer* geral se encontra no nível mais alto conectando diferentes Intranets, as quais possuem uma topologia hierárquica. A Figura 15 (b) mostra outro exemplo de topologia híbrida, na qual tem-se topologia *peer-to-peer* geral dentro de cada grupo e uma topologia *peer-to-peer* acíclica em um nível mais elevado. Topologias híbridas são mais difíceis de serem gerenciadas, pois se faz necessário ter conhecimento a priori da estrutura

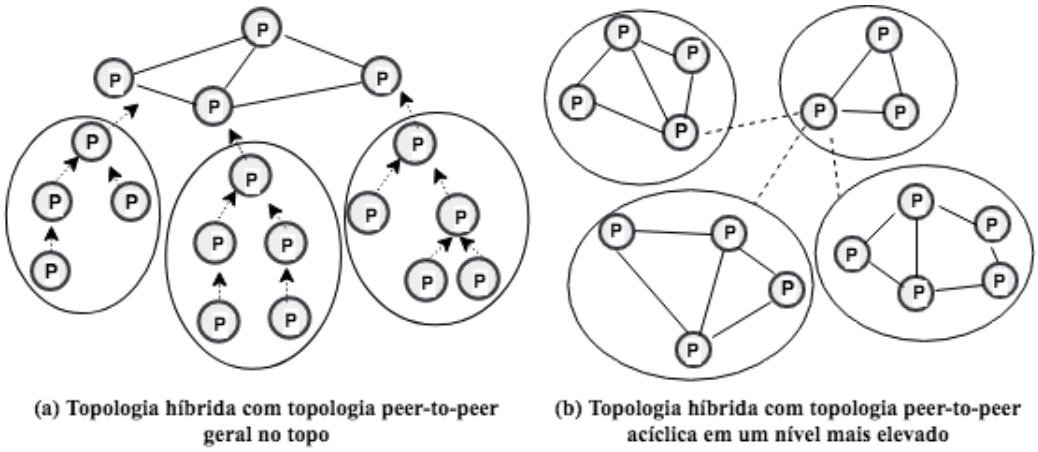
Figura 14 – Topologias: a) Hierárquica; b) *Peer-to-peer* acíclica; c) *Peer-to-peer* geral.



Fonte: (CARZANIGA; ROSENBLUM; WOLF, 2001)

do serviço para realizar uma subdivisão de *peers* e hierarquias. Além disso, o roteamento acaba tornando-se mais complexo.

Figura 15 – Exemplos de topologias híbrida.



Fonte: (CARZANIGA; ROSENBLUM; WOLF, 2001)

Alguns serviços de notificação de eventos conectam todos os *brokers* (roteadores em nível de aplicação) em uma única rede de forma hierárquica formando uma árvore (p. e., (COSTA et al., 2007), (LAI; CAO; ZHENG, 2009) e (SILVA et al., 2014)). A árvore de *brokers* é explorada para o encaminhamento de mensagens (i. e., subscrições e eventos). Normalmente, essas mensagens não são difundidas para a árvore inteira, mas encaminhadas em direção aos nós interessados de acordo com o conteúdo da mensagem e as subscrições armazenadas nos nós da árvore. Outros sistemas conectam *brokers* em uma topologia

peer-to-peer acíclica (p. e., (MEIER; CAHILL, 2002) e (CARZANIGA; ROSENBLUM; WOLF, 2001), *peer-to-peer* geral (p. e., (SOUTO et al., 2006)) e ainda de forma híbrida.

3.6 Roteamento de Eventos

O roteamento de eventos é o processo de entregar um evento para todos os assinantes que emitiram uma subscrição correspondente antes da publicação (BALDONI; VIRGILLITO, 2005). Normalmente, esse processo é realizado sempre depois do processo de *matching* em cada participante envolvido. Uma visita a todos os nós em um serviço de notificação é realizada para encontrar, para qualquer evento publicado, todos os assinantes cuja subscrição registrada está presente no serviço de eventos no momento da publicação. Desta forma, uma vez que a organização e a topologia dos participantes são definidas, caminhos de roteamento apropriados são estabelecidos para assegurar que as notificações publicadas por um publicador serão corretamente entregues para todos os assinantes que se subscreveram para elas (CARZANIGA; ROSENBLUM; WOLF, 2001). Três são as categorias identificadas de roteamento de eventos (BALDONI; VIRGILLITO, 2005): (1) Algoritmos de *flooding*: *event flooding* e *subscription flooding*; (2) Algoritmos seletivos: baseados em filtro ou conteúdo e baseados em *Rendezvous*; (3) Algoritmos *gossiping* de eventos. Nas próximas seções os algoritmos serão apresentados com maior aprofundamento em relação aos seguintes itens (BALDONI; VIRGILLITO, 2005):

1. ***Overhead* de mensagens**: o *overhead* introduzido na rede por enviar mensagens de subscrição e publicação. Medido pelo número de nodos que são atravessados pelo evento ao longo da propagação. Um algoritmo de roteamento pode alcançar todos os assinantes em um único salto; entretanto, todas as mensagens a mais são consideradas como *overhead*.

2. ***Overhead* de memória**: a quantidade de informação armazenada em cada processo. Relacionada com a replicação de subscrições, que são o número de cópias de cada subscrição que são apresentadas no sistema.

3. ***Overhead* de cálculo**: o tempo de filtragem como um todo em cada *broker* ou participante do serviço de notificação de eventos utiliza para realizar o *matching*.

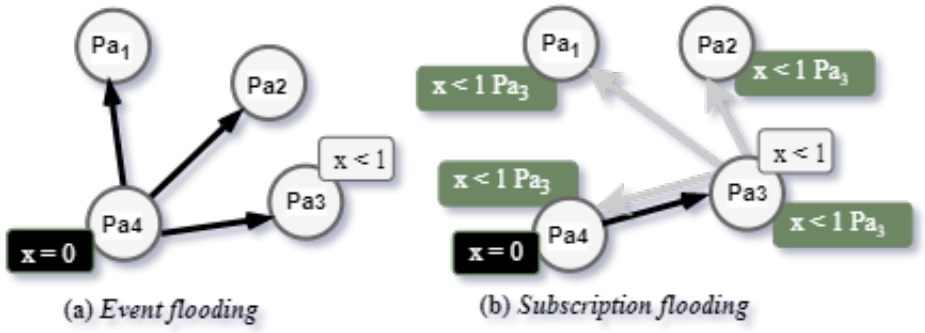
Cabe salientar que em qualquer sistema distribuído, uma árvore de disseminação de eventos deve ser dinamicamente construída para que eventos possam ser roteados dos publicadores para todos os assinantes interessados (OZALP; TEKIN, 2014). Essa árvore pode ser criada através de uma rede *overlay* de roteamento (*overlay routing network*). Nem sempre essa rede *overlay* de roteamento espelha a topologia real da rede subjacente. A escolha do algoritmo de disseminação de eventos, algoritmo de roteamento de eventos,

determina a escalabilidade global de um *middleware* baseado em eventos (OZALP; TEKIN, 2014).

3.6.1 Algoritmos de Roteamento Event flooding e Subscripition flooding

No algoritmo *event flooding* (inundação de eventos) cada evento é propagado do publicador para todos os participantes do serviço de notificação de eventos. Cada participante recebe e encaminha o evento para todos os participantes conhecidos. Com isso há um grande *overhead* de mensagens na rede (enviam para todos os participantes menos daquele que recebeu) levando à saturação da rede (CHAND; FELBER, 2003); entretanto, pouco *overhead* de memória, pois nenhuma informação de roteamento precisa ser armazenada na memória. A filtragem, aplicação do *matching*, ocorre no lado consumidor, ou seja, no assinante. A Figura 16 (a) apresenta um exemplo desse algoritmo, na qual o evento $x = 0$ é publicado para todos os nós, mas somente o assinante P_{a3} está inscrito para o mesmo, o qual recebe e trata o evento. Na Figura 16, o participante é denotado por P_a, as caixas pretas e as setas pretas representam os eventos publicados e enviados, respectivamente; as caixas brancas e as setas cinza representam subscrições armazenadas e enviadas, respectivamente; as caixas verdes representam a informação de roteamento armazenada.

Figura 16 – Exemplos de algoritmos de eventos (a) *Event flooding* e (b) *Subscription flooding*.



Fonte: (BALDONI; VIRGILLITO, 2005)

Na abordagem *subscription flooding* (inundação de subscrições), cada subscrição é enviada para todos os participantes juntamente com o identificador do assinante (Figura 16 (b)). Da mesma forma que o *event flooding*, cada participante recebe e encaminha para todos os participantes conhecidos a subscrição recebida. Assim, todos os participantes terão o conhecimento de todas as subscrições. Desta forma, assinantes podem ser alcançados em um único salto e eventos não interessantes podem ser filtrados nos publicadores. Na Figura 16 (b), a subscrição para o evento $x < 1$ é enviada do participante P_{a3} para todos

os participantes. Observe que a subscrição para o evento $x < 1$ possui a identificação do assinante (i. e., P_{a3}), a qual é armazenada em todos os participantes que a receberam. Desta forma, o publicador P_{a4} faz uma filtragem e envia o evento $x < 0$ somente para o assinante interessado, a saber, o assinante P_{a3}.

No algoritmo *subscription flooding* há um *overhead* significativo de mensagens na rede em relação as subscrições, além de *overhead* de memória, uma vez que informações de subscrições precisam ser armazenadas nos participantes.

3.6.2 Roteamento de Eventos Seletivo

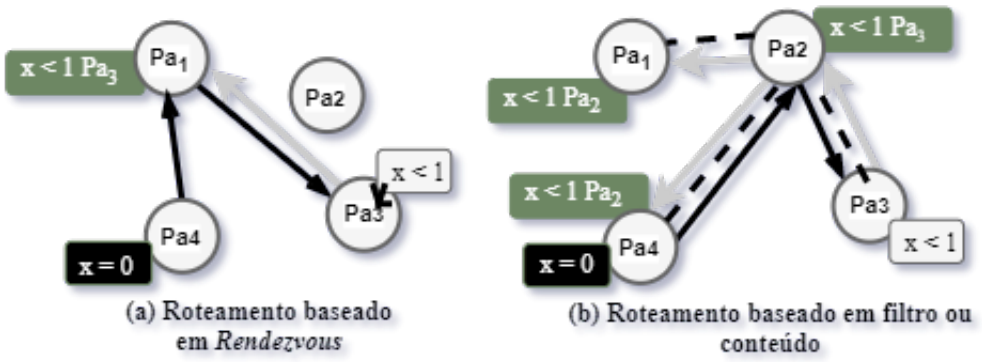
O princípio dos algoritmos de evento seletivo é reduzir o *overhead* de mensagens do algoritmo *event flooding* por selecionar somente um subconjunto de participantes armazenando cada subscrição e um subconjunto de participantes sendo visitados em cada evento. Os participantes são configurados para comparar os eventos com todas as subscrições e calcular uma lista de destinos para o encaminhamento dos eventos. Essa abordagem é chamada de *match-first* (CHAND; FELBER, 2003).

3.6.2.1 Roteamento baseado em *Rendezvous*

O roteamento Rendezvous (*Rendezvous-based Routing*) é baseado em duas funções: SN – associa subscrição a participantes e EN - associa eventos a participantes no serviço de eventos. Dada uma subscrição α, $SN(\alpha)$ retorna um conjunto de participantes, nomeados nodos *Rendezvous* da subscrição α, os quais são responsáveis para armazenar a subscrição α e encaminhar os eventos correspondentes a subscrição α para todos os assinantes daquela subscrição α. A função $EN(e)$ complementa SN por retornar os nodos *Rendezvous* de E, os quais são nodos responsáveis por fazer a correspondência entre o evento e todas as subscrições para receber aquele evento no sistema. Mediante a emissão da subscrição α, um assinante envia a subscrição α para os nodos em $SN(\alpha)$, os quais armazenam α e o identificador do assinante. Todas as subscrições para um mesmo evento podem ser armazenadas em um mesmo participante. A entrega de eventos é simplificada, consistindo na criação de uma arvore de difusão com uma única raiz iniciando dos *brokers* e espalhando-se a todos os assinantes. Na Figura 17 (a), a subscrição $x < 1$ é enviada do participante P_{a3} ao participante P_{a1}. SN associa a subscrição $x < 1$ ao nodo P_{a3}. O participante P_{a1}, nodo *Rendezvous*, por sua vez armazena a subscrição e envia eventos que correspondem a subscrição a todos os seus assinantes, nesse caso P_{a3}. Assim, P_{a1} recebendo o evento $x = 0$, envia-o ao participante P_{a3} que está inscrito para recebê-lo. A função EN, neste caso, retornou para a subscrição $x < 1$ o nodo P_{a1}, o qual realizou o *matching* entre o evento e o participante interessado em receber o evento.

Neste tipo de roteamento, existe um *overhead* de memória devido ao armazenamento das subscrições.

Figura 17 – Exemplos de algoritmos de eventos *Rendezvous* e baseado em filtro.



Fonte: (BALDONI; VIRGILLITO, 2005)

3.6.2.2 Roteamento baseado em Filtro ou Conteúdo

No roteamento baseado em filtro ou conteúdo (Filtering-based Routing or Content-based Routing - CBR), os eventos são encaminhados com base em seu conteúdo e nos interesses expressos por meio de subscrições realizadas pelos assinantes do serviço de eventos (CHAND; FELBER, 2003). A entrega dos eventos aos assinantes é gerenciada inteiramente pelo seu conteúdo (MAJUMDER et al., 2009), (OZALP; TEKIN, 2014). Isso significa que os caminhos de roteamento são determinados pelo conteúdo. Assim, quando um evento chega a um participante ele é comparado com as subscrições armazenadas e é encaminhado aos assinantes interessados. Isto quer dizer que o roteamento considera os valores dos atributos das mensagens de subscrições e eventos. As semânticas das requisições para eventos são amplas, sendo o conteúdo descrito com várias tags permitindo subscrições relacionadas a qualquer conteúdo descrito com aquelas tags. Desta forma, o roteamento baseado em conteúdo é muito diferente da forma de roteamento tradicional em que as mensagens são roteadas com base no endereço IP do destino. No roteamento baseado em conteúdo é o assinante que determina a entrega das mensagens, não o publicador.

Uma característica interessante do roteamento baseado em filtro ou conteúdo se dá em relação ao encaminhamento dos eventos, eventos são encaminhados somente aos participantes que se situam no caminho da rede em que existem subscrições para aqueles eventos. Com isso, os eventos são notificados aos assinantes interessados. A informação dos caminhos de difusão requer informação de roteamento que necessita ser armazenada e mantida nos participantes. A informação de roteamento no participante é associada a cada um de seus vizinhos e consiste de um conjunto de subscrições que são alcançáveis através de um participante que pode ser um *broker* (intermediário). Cada *broker* mantém três estruturas (BALDONI; VIRGILLITO, 2005): uma lista de vizinhos, uma tabela de roteamento e uma lista de subscrições. A tabela de roteamento associa um vizinho com

uma entrada representando um conjunto de subscrições. As tabelas de roteamento são configuradas adequadamente por cada *broker* para que as mensagens publicadas possam ser notificadas aos assinantes interessados (ZHAO; WU, 2013). A lista de subscrições associa um participante à sua subscrição. A função *match* combina um evento com a lista de subscrições ou uma entrada na tabela de roteamento e retorna uma lista com todos os participantes correspondentes, os quais são os participantes interessados no evento. No roteamento baseado em filtro ou conteúdo, o *overhead* de mensagens é reduzido pela identificação de eventos que os assinantes não estão interessados, os quais não são encaminhados. Existe um *overhead* de memória devido ao armazenamento das três estruturas de controle em cada participante. A Figura 17 (b) ilustra esse tipo de roteamento. O assinante P_{a3} envia uma subscrição para o evento $x < 1$ ao seu vizinho que é o participante P_{a2}. O participante P_{a2} recebe a subscrição e envia para todos os seus vizinhos (isto é, P_{a1} e P_{a4}). O participante P_{a4} verifica que possui o evento para o qual a subscrição foi realizada e que o caminho para que o evento chegue a P_{a3} é P_{a2}, enviando assim o evento correspondente. O participante P_{a2} recebendo o evento encaminha-o para o assinante P_{a3}. A filtragem ocorreu nos participantes P_{a2} e P_{a4}.

Como os assinantes definem o tipo de eventos que eles estão interessados, os publicadores não precisam manter a localização da população dos assinantes e podem simplesmente injetar eventos na rede. O evento determina como se dará o encaminhamento dos eventos de acordo com os interesses dos participantes (MOTTOLA; CUGOLA; PICCO, 2008). Dessa forma, assinantes com recursos escassos, como é o caso de dispositivos móveis com largura de banda limitada, podem restringir o tipo e a quantidade de dados que eles recebem por fazer subscrições altamente seletivas e, assim limitar seu tráfego de entrada na rede.

Por fim, no roteamento baseado em conteúdo ou filtro os consumidores definem suas próprias classes de mensagens. Isso significa que o consumidor seleciona somente as mensagens de seu interesse por meio de suas subscrições. Ainda, no roteamento baseado em conteúdo e no roteamento *Rendezvous*, o tempo de filtragem nos participantes é maior que nos outros tipos de algoritmos, não se apresentando escalável quando o número de subscrições aumenta.

3.6.2.3 Roteamento baseado em Gossip

Nos protocolos baseados em *gossip* (roteamento epidêmico), cada participante entra em contato com um ou alguns participantes em cada instante e troca informações com esses participantes (COSTA et al., 2003). São protocolos simples e não requerem manter estrutura dos dados do roteamento de eventos em cada nodo. O gargalo dessa abordagem é um *overhead* de mensagens devido à redundância de mensagens em cada instante.

3.7 Trabalhos Relacionados a Middlewares baseados em Eventos

Vários *middlewares* baseados em eventos foram analisados neste trabalho, buscando entender sua arquitetura (isto é, quais são seus componentes, suas propriedades e relacionamentos entre os componentes), método de subscrição de eventos, qual o modelo de mensagens é utilizado e como acontece o roteamento de mensagens.

SIENA (CARZANIGA; ROSENBLUM; WOLF, 2001) é um serviço de notificação de eventos de propósito geral projetado para funcionar sobre a Internet. Logo, apresenta uma arquitetura distribuída. A subscrição é baseada em conteúdo e SIENA é implementado como um conjunto de servidores distribuídos com publicadores e assinantes que publicam e recebem eventos do serviço de eventos, respectivamente. As topologias hierárquica, *peer-to-peer* acíclica com encaminhamento de subscrições e *peer-to-peer* acíclica com encaminhamento de anúncios foram implementadas. Quanto ao roteamento de eventos, SIENA utiliza roteamento baseado no conteúdo dos eventos. Para isso, anúncios são utilizados para gerar caminhos para as subscrições, os quais estabelecem os caminhos dos eventos. Cada anúncio é propagado através da rede formando uma árvore que alcança cada servidor. Quando um servidor recebe uma subscrição, ele propaga a subscrição ao contrário ao longo dos caminhos para todos anunciadores que submeteram anúncios relevantes, assim ativando aqueles caminhos. Quando um publicador publica um evento que corresponde a uma subscrição, o evento é roteado em direção ao assinante seguindo o caminho reverso realizado pela subscrição. SIENA também permite a identificação de padrões entre os eventos. Enquanto um filtro é comparado a um único evento com base nos valores dos atributos do mesmo, um padrão é comparado a um ou mais eventos com base na combinação que formam e nos valores dos atributos. Em sua forma mais genérica, um padrão pode correlacionar eventos de acordo com qualquer relação. Embora SIENA tenha filtro de eventos e junção de eventos para notificação dos assinantes, não permite o envio de eventos somente aos assinantes mais adequados.

Hermes (PIETZUCH; BACON, 2002) é um *middleware* baseado em eventos, sendo composto por *events client* (publicadores de eventos e assinantes de eventos) e *event brokers* (o *middleware* propriamente dito). Os *events client* utilizam o serviço oferecido pelo *middleware* para se comunicar utilizando eventos. Por outro lado, os *event brokers* aceitam inscrições de assinantes e entregam os eventos publicados para todos os assinantes interessados. A subscrição para eventos se dá através da combinação dos modelos de subscrição baseado em tipo e conteúdo. Primeiramente, o assinante do evento especifica o tipo de evento em que está interessado e então fornece uma expressão filtro que opera nos atributos do tipo de evento. De qualquer forma, pode ser visualizada como uma subscrição baseada em conteúdo. A arquitetura do Hermes segue uma abordagem em camadas e ele é implantado sobre uma camada de rede semelhante à camada de rede da Internet. Em relação à topologia, a topologia *peer-to-peer* geral, ou com ciclos é utilizada. Hermes

utiliza o algoritmo de roteamento seletivo baseado em *Rendezvous* para o roteamento de eventos apresentando duas variações deste, o roteamento baseado em Rendezvous baseado em tipo e o roteamento baseado em *Rendezvous* baseado em tipo e conteúdo. Com isso, o *middleware* faz uma espécie de junção de subscrições, encaminhando-as a um próximo nó somente se a subscrição for mais geral que aquelas que ele possui armazenado. Neste *middleware* não é possível realizar seleção dos melhores assinantes para o envio de eventos, bem como ir controlando a quantidade de assinantes que recebem o evento, quais os mais próximos e menos ou mais fatigados e os mais distantes e menos ou mais fatigados para atendimento de uma emergência, por exemplo.

Steam (MEIER; CAHILL, 2002) é um serviço de eventos desenvolvido para distribuição de eventos entre entidades (i. e., publicadores e assinantes) que residem em dispositivos de computação móvel que utilizam redes sem fio para interagir. Foi projetado para o domínio de uma aplicação de gerência de tráfego no qual os assinantes são nós móveis (carros e ambulâncias) que se movimentam em uma mesma área física dos publicadores (p. e., semáforos, carros) e compartilham eventos a respeito da atual situação de tráfego. As entidades do *middleware* interagem com quem está próximo delas. Os publicadores definem a proximidade geográfica para seus eventos serem entregues somente aos assinantes localizados fisicamente naquela região. STEAM considera que todas as entidades podem estimar sua própria localização. Entretanto, antes de publicar um evento, mensagens de anúncio são enviadas com o tipo do evento, o qual contém informações de proximidade. Assinantes, por sua vez, criam subscrições baseadas no tópico do evento e o *middleware* entrega os eventos somente aos assinantes localizados em sua proximidade, conforme mencionado. A arquitetura de STEAM é *peer-to-peer* na qual os nós móveis agem como publicadores e assinantes dos eventos, ou seja, as interações não envolvem qualquer infraestrutura central ou distribuída. Logo, a topologia é dinâmica sem uma organização pré-definida. O algoritmo de eventos adotado é uma espécie de algoritmo epidêmico com filtros presentes nos publicadores e nos assinantes dos eventos. Mesmo os publicadores utilizando proximidade geográfica para envio de eventos não resolve a questão do envio somente aos assinantes mais adequados daquela região. Se o envio ocorre somente para aqueles assinantes mais próximos, nem sempre serão os mais adequados e pode ser que os disponíveis não serão em número suficiente para atender aquele evento, especialmente se situações de emergência forem consideradas. Além disso, a realização de subscrição por proximidade não ajuda neste sentido e outros fatores devem ser considerados.

SensorBus (RIBEIRO et al., 2005) é um *middleware* orientado a mensagens que emprega o paradigma *publish-subscribe* para Redes de sensores sem fio. Sua arquitetura é centralizada. O *middleware* oferece facilidade para o desenvolvimento de aplicações eficientes energeticamente e trata características das Redes de sensores permitindo a livre troca do mecanismo de comunicação entre os nós sensores. Os nós sensores podem ser endereçados por seus próprios atributos (único endereçamento) ou por atributos extraídos

do ambiente físico (nomeação baseada em atributos). O desenvolvimento de uma aplicação utilizando SensorBus consiste em codificar as partes de um consumidor e produtor. O código consumidor executa na máquina usuário enquanto o código produtor executa nos nós sensores. Inicialmente, um barramento para comunicação entre produtores e consumidores é criado. Posteriormente, produtores e consumidores são criados, bem como canais para comunicação com o barramento. O produtor gera itens de dados e coloca-os em canais enquanto o consumidor encontra e "tritura" esses dados. O método de busca em SensorBus é *pull* e o canal pode ser dividido entre produtores e consumidores. O *middleware* não permite entrega para somente um grupo de assinantes e sua arquitetura não é adequada para Cidades Inteligentes.

EMMA (MUSOLESI; MASCOLO; HAILES, 2006) é um *middleware* orientado a mensagens, mais especificamente uma adaptação do *Java Message Service* (JMS), projetado para ambientes móveis. Os modelos de comunicação ponto a ponto e *publish-subscribe* foram implementados no *middleware* proposto. No modelo *publish-subscribe* implementado, alguns nós armazenam subscrições e tratam tópicos. As subscrições são baseadas em tópicos, podem ser duráveis ou não duráveis. Nas assinaturas duráveis, se o assinante se desconectar, eventos não são armazenados, mas são enviados utilizando o protocolo epidêmico. Assinaturas duráveis permanecem válidas durante as possíveis desconexões do assinante. Possuem um tempo de expiração predefinido no caso dos assinantes se movimentarem e saírem da faixa de contato. Se a assinatura não for durável, a assinatura é deletada quando o assinante é desconectado, ou melhor dizendo, durante a desconexão eventos não são enviados utilizando o protocolo epidêmico. Se o assinante se torna acessível novamente, deve realizar outras assinaturas para receber os eventos correspondentes. Mensagens de cancelamento de uma inscrição são enviadas da mesma forma que mensagens de assinatura para eventos. Ainda, tópicos são anunciados por meio de um registro (*registry*) e um assinante querendo assinar o tópico registra-se com o produtor contendo o tópico. Quando um produtor quer enviar um evento para a lista de tópicos, ele envia o evento para o tratador do tópico que pode ser ele mesmo. O tratador do tópico então encaminha o evento para todos os assinantes de forma síncrona (quando o assinante estiver conectado) ou utilizando o protocolo epidêmico (quando o assinante estiver desconectado) se as subscrições forem duráveis, no último caso. Quando um evento é entregue a um dos assinantes, esse recebedor é deletado da lista assim que confirmar o recebimento do evento. As subscrições podem ser associadas a prioridades. A arquitetura do serviço de eventos é *peer-to-peer* e a topologia não é definida. O que é importante observar é que quando o algoritmo epidêmico é utilizado a filtragem acontece no assinante. Isso se o assinante receber o evento antes de realizar uma nova subscrição com o tratador do tópico interessado. Um mecanismo interessante para minimizar o *overhead* de armazenamento nos nós que armazenam as subscrições é a deleção das mesmas quando os assinantes confirmam o recebimento dos eventos. Entretanto, dependendo do contexto não é indicada. Outra

questão é que mesmo utilizando priorização no envio de eventos como EMMA faz, não é garantido que os melhores assinantes irão receber o evento. A priorização só serve para enviar uma mensagem antes de outra. Além disso, a arquitetura utilizada é *peer-to-peer*, não sendo indicada para um cenário de grande extensão geográfica como uma cidade.

Mires (SOUTO et al., 2006) é um *middleware* que implementa a comunicação *publish-subscribe* para aplicações de Redes de sensores sem fio. A comunicação entre os nodos sensores desse ambiente consiste de três fases: (i) os nós sensores na rede publicam seus tópicos disponíveis (p. e., temperatura e umidade) coletados dos sensores locais; (ii) as mensagens anunciadas são encaminhadas ao nó *sink* utilizando um algoritmo de roteamento. Uma aplicação conectada ao nó *sink* poderia, depois disso, assinar os tópicos desejados a serem monitorados (i. e., realizar uma seleção através de uma interface gráfica); (iii) as mensagens de subscrição são difundidas em *broadcast* para os nós sensores na rede. Depois de receber os tópicos assinados, os nós podem publicar seus dados coletados (i. e., os eventos) para a rede que os encaminha ao nó *sink*. Posteriormente, os eventos são entregues para a aplicação. Desta forma, as subscrições para eventos são realizadas através de tópicos. O *middleware* apresenta uma arquitetura em camadas e é distribuído. Para o roteamento *multi-hop* foi estabelecida uma rede com topologia hierárquica em árvore. O *middleware* foi utilizado em uma aplicação de monitoração de um ambiente. Este *middleware* não permite seleção dos melhores assinantes disponíveis.

Publish/Subscribe Notification middleware for Vehicular networks - PSNMVN (LEONTIADIS, 2007) é o projeto de um *middleware publish-subscribe* para ser utilizado em Redes veiculares. Os autores almejaram combinar as vantagens de comunicação do uso de uma infraestrutura de comunicação fixa e outra *ad-hoc*. Os veículos são considerados como sensores móveis que coletam informações sobre as condições de tráfego, acidentes, entre outros. Assim, assumem o papel de publicadores, mas também podem ser os interessados nos dados coletados (i. e., os assinantes). Nas subscrições, os veículos indicam o interesse sobre certos tipos de notificações se inscrevendo em tópicos com uma informação adicional em relação ao ponto de interesse do evento. As subscrições são armazenadas localmente no assinante para realização do *matching* no assinante quando ele recebe o evento. Os veículos reportam as informações coletadas para uma estação base mais próxima ou então para um *hopspot Wifi* utilizando roteamento veículo para veículo ou conectando-se diretamente com a estação base. A notificação que é gerada no publicador define a área de interesse e a área de disseminação da notificação. Desta forma, um sistema pode combinar as informações obtidas das várias fontes e gerar alertas de tráfego a respeito das áreas afetadas. Inicialmente, os alertas são enviados as estações mais próximas. Delas, são roteados aos segmentos da rodovia afetados utilizando comunicação veículo a veículo. Ao chegar à área afetada, as notificações são enviadas em *broadcast* para os veículos afetados, ou seja, um algoritmo de pesquisa é utilizado para descobrir os vizinhos mais próximos para o encaminhamento do evento. Os veículos são considerados como estações móveis

que levam as réplicas do evento para disseminação durante o tempo em que forem válidos. Quando um veículo recebe um evento, ele aplica um filtro. Logo, o sistema de navegação do veículo que recebe esse evento pode avaliar as informações fornecidas (se estiverem interessadas ou não) e recalcular automaticamente uma rota, evitando as áreas afetadas. Os autores mencionam que o *middleware* pode ser expandido para suportar inúmeras aplicações como avisos de acidentes, obras rodoviárias, vagas de estacionamento gratuitas, preços de combustível, anúncios, entre outros. Embora o *middleware* possa enviar eventos para veículos que estejam somente dentro de uma área específica não permite a seleção entre os assinantes interessados mais aptos dessa área. Além disso, o *middleware* trabalha com filtros no lado do assinante dificultando ainda mais esta tarefa.

RUNES (COSTA et al., 2007) é um *middleware publish-subscribe* projetado para um cenário de desastre em um túnel rodoviário (p. e., fogo ou derramamento de produtos químicos). O túnel é equipado com sensores em rede, atuadores, e dispositivos maiores e mais potentes. Os dispositivos mais potentes agem como *gateways* e permitem aos sensores reportar leituras monitoradas diretamente para os sistemas atuadores e o centro de controle do túnel. Em uma emergência, os bombeiros entram no túnel em grupo e o sistema oferece informações aos bombeiros que se movimentam dentro do túnel. O *middleware* trata da questão da heterogeneidade, escassez de recursos e dinamismo da rede. A subscrição para eventos é entendida como baseada em conteúdo. A arquitetura é distribuída e a topologia é hierárquica com roteamento *multi-hop*. RUNES é um *middleware* baseado em eventos construído para uma aplicação específica. Os eventos são enviados somente aos bombeiros que expressaram interesse no evento não apresentando possibilidade de envio aos assinantes mais apropriados, por exemplo, os bombeiros mais próximos ao incidente.

PSWare (LAI; CAO; ZHENG, 2009) é um *middleware publish-subscribe* projetado e implementado para Redes de sensores sem fio que permite a definição de relações temporais entre diferentes eventos, chamados de eventos compostos. Por exemplo, a colisão entre dois carros precisa ser descrita pela velocidade dos dois carros e a distância entre eles. Uma linguagem de definição de eventos (*Event Definition Language* - EDL) que permite aos programadores de aplicação especificarem subscrições para eventos de forma simples foi definida. Um compilador para compilar as subscrições (programas escritos em EDL em *bytecodes*), um ambiente em tempo de execução nos nós sensores para executar os *bytecodes* e um protocolo de detecção de eventos compostos visando eficiência energética foram desenvolvidos. As subscrições para eventos podem ser realizadas através de tópicos e conteúdo, ou seja, para eventos primitivos (evento com um único tópico e seu valor correspondente) e eventos compostos. A arquitetura é distribuída, embora que os sensores sejam implantados em um local, a topologia dos nós sensores é hierárquica e o roteamento, embora não explicito, é *multi-hop*. O *middleware* pode ser utilizado para aplicações como monitoramento ambiental e rastreamento de objetos. Embora os autores mencionam que o diferencial do *middleware* é a possibilidade de expressar eventos compostos que possuem

uma relação temporal entre os mesmos, não deixa de ser uma subscrição baseada em conteúdo que utilizada uma linguagem de expressão muito parecida com SQL para ser realizada. A maioria dos que foram estudados até agora permitem expressar eventos de forma composta, só de um jeito diferente. Também neste *middleware* não ó possível realizar a escolha dos assinantes mais adequados.

TinyDDS (BOONMA; SUZUKI, 2012) é um *middleware publish-subscribe* para Redes sem fio que implementa uma especificação de protocolo *publish-subscribe* da OMG. O *middleware* se preocupa em oferecer interoperabilidade entre redes de acesso e Redes de sensores sem fio, agregação de dados e roteamento de eventos. A subscrição para eventos se dá através de tópicos e para o roteamento de eventos, três protocolos podem ser configurados para trabalhar com o *middleware*: roteamento baseado em *spanning tree*, *Distributed Hash Table* - DHT ou Moonson. Várias RSSFs podem ser utilizadas, sendo um *gateway* DDS utilizado para interoperabilidade entre as Redes de sensores sem fio e a Internet. O *gateway* é uma aplicação Java que interage com o *middleware* TinyDDS, que executa nos nós sensores. Quando um cliente DDS assina um tópico, a informação de subscrição é distribuída sobre a rede de acesso sobre DDS. Como consequência, o *gateway* DDS pode interceptar a informação de subscrição e armazená-la em uma lista de subscrições. Assim, quando um nó sensor publicar um evento, o evento é distribuído na rede de sensores e interceptado pelo gateway DDS. Então, se o tópico do evento publicado corresponder com um tópico da lista de subscrições, ele é enviado aos assinantes interessados de acordo com o protocolo configurado. O cliente Web DDS executa em computadores pessoais ou dispositivos móveis. Ele pode comunicar-se com o *gateway* DDS para assinar os dados publicados pela rede de sensores mostrando o resultado no Google Maps. TinnyDDS é configurável e interoperável, sendo possível configurá-lo para operar com um protocolo específico de roteamento dentre três disponíveis. O roteamento baseado em *spanning tree* se ocupa em criar um caminho mínimo entre os nós da Rede de sensores sem fio quando a entrega de eventos acontece, mas não garante que os melhores assinantes recebam o evento. O roteamento baseado em DHT cria um canal lógico entre o nó *Rendezvous* e os publicadores e assinantes dos eventos. Também não permite seleção dos assinantes mais aptos. Moonson, por outro lado, embora permita a um nó sensor se adaptar a modificações e falhas na rede, não permite escolhe seletiva dos melhores assinantes.

PRISMA (SILVA et al., 2014) é um *middleware publish-subscribe* orientado a recursos para Redes de sensores sem fio. Assim, a rede rede pode ser um único recurso ou cada nó individual pode ser um recurso, bem como em cada nó muitos recursos podem existir (p. e., temperatura, luz). Para isso, PRISMA adota REST (*REpresentational State Transfer*), o qual trabalha com "coisas" na Web que são representadas como recursos, para facilitar o acesso aos dados da rede e suportar interoperabilidade entre o *middleware* e outras redes (o serviço de comunicação fica responsável por isso). A comunicação é oferecida

por um *WebService* e um *broker* (intermediário) que oferece comunicação assíncrona. Interfaces REST interagem com as aplicações cliente e o *middleware* adota o paradigma *publish-subscribe* para notificar os clientes sobre eventos de interesse. Para receber as mensagens de notificação, os clientes precisam se inscrever em tópicos de interesse. Um tópico *publish-subscribe* é um canal de comunicação assíncrono usado pelo *middleware* para publicar mensagens de interesse aos assinantes. Esse tópico pode ser criado se os requisitos da aplicação podem ser satisfeitos com os recursos da Rede de sensores sem fio. Se a rede pode reunir os requisitos, o cliente pode receber o tópico em resposta a uma requisição REST. Por outro lado, pode receber uma mensagem de erro. Com essa resposta, o cliente pode assinar o tópico desejado com o *broker* e receber os dados de interesse quando eles estiverem disponíveis. Desta forma, as subscrições são realizadas por tópicos e a arquitetura é centralizada. Entretanto, não é possível para este *middleware* o envio seletivo de eventos.

Apache Kafka (KAFKA, 2018) é uma plataforma de *streaming* distribuída que permite enviar mensagens entre publicadores e assinantes através de tópicos. Quando um produtor gera uma mensagem sobre o tópico, os consumidores desse tópico são notificados com uma cópia da mensagem publicada. Os tópicos são divididos em várias partições distribuídas em servidores. A comunicação é realizada com o protocolo de controle de transporte (*Transport Control Protocol* - TCP). Todos os assinantes do tópico recebem a notificação do evento. Contudo, uma seleção dos assinantes mais apropriados não é permitida.

RabbitMQ (PIVOTAL, 2019) é um *broker* de mensagens de código aberto que emprega vários padrões de comunicação como ponto a ponto, *request/reply* e *publish-subscribe* suportando vários protocolos de comunicação. Foi desenvolvido originalmente para implementar o protocolo *Advanced Message Queuing Protocol* - AMQP, todavia implementa outros protocolos como *Simple Text Oriented Messaging Protocol* - STOMP, *Message Queue Telemetry Transport* - MQQT e HTTP (utilizado por RabbitMQ para transmitir mensagens sobre ele). AMQP é um protocolo *publish-subscribe* da camada de aplicação para troca de mensagens entre dois processos sobre uma rede (STANDARD, 2012). O protocolo foi projetado para uso sobre um protocolo de transporte confiável, como o TCP. Assim, oferece entrega de dados de forma confiável. Uma rede AMQP consiste de um conjunto de nós (clientes e *brokers*) conectados por meio de *links*. Nós são responsáveis por armazenamento confiável e/ou entrega de mensagens. Mensagens podem originar-se de um nó, terminar nele ou passar por vários nós. A comunicação em AMQP é realizada por meio de *exchanges* e filas de mensagens (*queues*) (AL-FUQAHA et al., 2015). *Exchanges* são usadas para rotear as mensagens para as filas apropriadas. O roteamento entre *exchanges* e as filas de mensagens é baseado em regras e condições pré-definidas. As mensagens são armazenadas em filas de mensagens e então entregues aos recebedores. O AMQP define uma camada de mensagens sobre a camada de transporte. As mensagens

AMQP são tratados nessa camada. MQTT é um protocolo de transporte de mensagens *publish-subscribe* do tipo cliente-servidor (STANDARD, 2014), o qual foi proposto pela IBM para comunicação via-satélite com equipamentos encontrados em campos de petróleo (MOHANTY; CHOPPALI; KOUGIANOS, 2016). O protocolo executa sobre TCP/IP ou sobre outros protocolos de transporte que ofereçam conexão bidirecional, ordenada e sem perda de pacotes. STOMP é um protocolo de mensagens baseado em texto, assíncrono, para troca de mensagens entre clientes através de servidores de mediação (MAGNONI, 2015). Muitos *brokers* de mensagens o implementam e bibliotecas de clientes. MQTT é um protocolo de comunicação assíncrona. Algumas terminologias utilizadas pelo protocolo são (STANDARD, 2014): (i) *Application Message* – os dados levados pelo protocolo MQTT através da rede para a aplicação; (ii) *Client* (cliente) – um programa ou dispositivo que utiliza MQTT. O cliente pode (através de uma conexão com um servidor) publicar mensagens (*publisher*) em que outros clientes possam estar interessados; subscrever-se para mensagens em que eles possam estar interessados em receber (*subscriber*); desconectar-se do servidor; (iii) *Server*: um programa ou dispositivo que age como um intermediário (*broker*) entre clientes que publicam mensagens (*publishers*) e clientes que realizam subscrições (*subscribers*). Um servidor aceita conexões de rede dos clientes, aceita mensagens publicadas por clientes, processa subscrições e requisições de cancelamento dos clientes, encaminha mensagens que correspondem as subscrições dos clientes (permite comunicação muitos para muitos). Ou seja, gerencia e roteia as mensagens entre dispositivos/aplicações IoT (PERERA et al., 2017). Em resumo, os *publishers* geram dados de interesse, os quais são transmitidos aos *subscribers* através dos *brokers*. Um dispositivo interessado pode registrar-se como um interessado em um tópico específico e ser informado pelo *broker* quando os *publishers* publicam tópicos de interesse. RabbitMQ utiliza um modelo de consumidor "ingênuo" e um *broker* inteligente entregando mensagens aos consumidores em um ritmo em que eles possam receber as mensagens. A comunicação no RabbitMQ pode ser síncrona ou assíncrona. Os publicadores enviam mensagens para *exchanges* e os consumidores recuperam as mensagens das *queues* (filas). Desacoplar os produtores das filas acontece por meio de *exchanges* e garante que os mesmos não sejam sobrecarregados com decisões de roteamento. RabbitMQ não permite escolha dos assinantes mais aptos para receber cada evento.

Por fim, a Tabela 3 mostra um comparativo entre os *middlewares* analisados.

Tabela 3 – Comparação entre *Middlewares Publish-Subscribe*.

Middleware	**Subscrição**	**Arquitetura e Topologia**	**Roteamento**	**Filtragem**	**Armazenamento de subscrições**	**Gerência de eventos**	**Domínio**
SIENA (SILVA et al., 2014)	Conteúdo	Centralizada-Distribuída (várias topologias implementadas)	Filtro ou conteúdo (seletivo)	Subconjunto de participantes, não nos publicadores, nem nos assinantes	Subconjunto de servidores	Subconjunto de servidores	Internet
Hermes (PIETZUCH; BACON, 2002)	Tipo e conteúdo	Distribuída com topologia *peer-to-peer* geral	*Rendezvous* e filtro (seletivo)	Subconjunto de *event brokers*, não nos publicadores, nem nos assinantes	Subconjunto de *event brokers*	Subconjunto de *event brokers*	Internet
Steam (MEIER; CAHILL, 2002)	Conteúdo	*Peer-to-peer* com topologia dinâmica	Epidêmico com filtros	Publicadores e assinantes	Publicadores	Publicadores e assinantes	Redes *ad-hoc*
SensorBus (RIBEIRO et al., 2005)	Conteúdo	Centralizada	-	Assinantes	Assinantes	Publicador	RSSFs
EMMA (MUSOLESI; MASCOLO; HAILES, 2006)	Tópico	*Peer-to-peer* com topologia dinâmica	Epidêmico	Publicadores e assinantes	Alguns participantes	Alguns participantes	Redes *ad-hoc*
Mires (SOUTO et al., 2006)	Tópico	Distribuída com topologia hierárquica	*Multihop*	Publicador	Em todos os nós	Em todos os nós	RSSFs
RUNES (COSTA et al., 2007)	Conteúdo	Distribuída com topologia hierárquica	*Multihop*	Publicador	Em todos os nós	Em todos os nós	RSSFs
TinnyDDS (BOONMA; SUZUKI, 2012)	Tópico	Distribuída	*DHT*, entre outros	Publicador	Em todos os nós	Em todos os nós	RSSFs
PSWare (LAI; CAO; ZHENG, 2009)	Conteúdo	Distribuída com topologia hierárquica	*Multihop*	Publicador	Em todos os nós	Em todos os nós	RSSFs

Middleware	**Subscrição**	**Arquitetura e Topologia**	**Roteamento**	**Filtragem**	**Armazenamento de subscrições**	**Gerência de eventos**	**Domínio**
PRISMA (SILVA et al., 2014)	Tópico	Centralizada	*Pull*	Assinantes	Publicadores e assinantes	Publicador	RSSFs
PSNMVN (LEONTIADIS, 2007)	Tópico	Distribuída com topologia dinâmica	*Event Flooding*	Assinante	Assinante	Em alguns nós	Rede com infraestrutura e *ad-hoc*
Apache Kafka (KAFKA, 2018)	Tópico	Distribuída	Conteúdo	Subconjunto de participantes	Subconjunto de participantes	Subconjunto de participantes	Internet
RabbitMQ (PIVOTAL, 2019)	Tópico	Distribuída	Conteúdo	Subconjunto de participantes	Subconjunto de participantes	Subconjunto de participantes	Internet

Fonte: Autoria Própria

4 O Modelo de Eventos *Event to Most Suitable Subscribers* - EMSS

O modelo de comunicação baseado em eventos com filtragem por aptidão de assinante EMSS foi projetado para ser utilizado sobre um protocolo de transporte confiável, como o *Transport Control Protocol* (TCP). Assim, oferece entrega de dados de forma confiável. EMSS foi proposto para *middlewares publish-subscribe* considerando uma arquitetura distribuída para o envio de eventos, no contexto de Cidades Inteligentes. Como consequência, para ser utilizado por sistemas de eventos que necessitem do envio de eventos para um subconjunto dos assinantes, dentre todos aqueles que deveriam receber o evento. Inicialmente, foi planejado para o atendimento às situações de emergência. Entretanto, é um modelo flexível que pode facilmente ser adaptado para outros serviços em Cidades Inteligentes, como por exemplo, um serviço de transporte ao passageiro.

EMSS, como um modelo de eventos, é um conjunto de regras que descreve a comunicação entre *brokers*, assinantes e o publicador de eventos. O *broker* é um processo de software que toma a decisão sobre quais assinantes devem receber e tratar o evento. Cada *broker* é responsável por uma região da cidade. O assinante recebe o evento e o trata da maneira mais adequada possível. O publicador representa o sistema de eventos (i. e., a aplicação) que utiliza o *middleware*. O papel e as funções de cada um deles são descritos suscintamente abaixo.

1. ***Broker***: Assume o papel de supervisor de uma região específica da cidade controlando todos os eventos que acontecem em sua área de abrangência. Desta forma, é responsável por todos os serviços disponíveis na cidade. De uma forma geral, desempenha as seguintes funções:

   a : Recebe solicitações para acionamento dos assinantes de um determinado serviço para tratar de um evento específico.

   b : Encaminha eventos aos assinantes necessários do respectivo serviço localizados em sua região de abrangência. Os assinantes necessários serão os que apresentarem as melhores condições para tratar o evento. Em uma situação de emergência, o melhor assinante pode ser àquela ambulância que tiver UTI móvel para atendimento da ocorrência.

   c : Encaminha mensagem para um *broker* específico solicitando o acionamento de um assinante localizado na região sob a supervisão daquele *broker*. O *broker* que recebe a mensagem, escolhe o assinante e o notifica. Se necessário, o assinante

se desloca até a região do primeiro *broker*, ou seja, à região em que o evento aconteceu.

d : Recebe subscrições enviadas por assinantes que entram em sua região de supervisão.

e : Detecta a saída de assinantes da sua região de abrangência.

Além disso, cada *broker* conhece o mapa da cidade, todos os demais *brokers* e todos os pontos de referência da cidade. Um ponto de referência pode ser uma quadra pública, um hospital, uma igreja, a casa de uma pessoa importante na cidade, por exemplo. Um evento sempre é associado a um ponto de referência e cada ponto de referência é usado como um guia para que o assinante chegue até o local do evento quando ele é ativado para tratá-lo. Além disso, o ponto de referência permite o cálculo do *fitness* (valor de aptição) antes que o evento aconteça, pois o *broker* pode proativamente ter os valores de *fitness* já a disposição quando um evento acontecer. O *fitness* é um valor calculado sobre as subscrições que é utilizado como base para o envio de eventos, segundo o modelo EMSS. Esse valor indica a adequação de um assinante para responder ao evento associado a um ponto de referência particular. É como base nesse valor que o *broker* escolher qual(is) assinante(s) devem ser notificados.

2. **Assinante**: Assume o papel de agente executor. Resumidamente, realiza as seguintes funções:

   a : Se subscreve para atendimento aos eventos de um determinado serviço.

   b : Recebe notificações de eventos do *broker* da região em que se encontra.

   c : Trata os eventos. Se necessário, deslocando-se até a localização do evento.

   d : Envia mensagens de vínculo ao *broker* da região em que se encontra para atualização de estado.

3. **Publicador**: Assume o papel de intermediário na ocorrência de um evento fazendo a interface com o *broker* da região onde acontece o evento para acionar os serviços adequados. Desempenha a função de comunicar o *broker* da região onde ocorre o evento para que os assinantes possam ser notificados de forma à tratar do evento.

Por fim, um serviço de notificação de eventos que permita a um sistema de eventos a notificação diferenciada de assinantes pode utilizar o modelo de eventos EMSS proposto neste trabalho. Conforme mencionado, o atendimento às emergências, como por exemplo, acidentes em meio de transporte, incêndio, salvamento, roubo, lesão corporal e o serviço de transporte de passageiros, semelhante ao serviço de TAXI e Uber, serão utilizados neste trabalho como estudo de caso na utilização do modelo de eventos EMSS. Assim, serão apresentadas algumas considerações acerca dos estudos de caso escolhidos.

- Em uma situação de emergência, o publicador é representado por um atendente das situações de emergência e em um serviço de transporte, pelo usuário que deseja fazer uso do serviço.
- No caso de uma situação de emergência, os *brokers* decidem sobre quais unidades móveis devem ser acionadas para atendê-la. Em um serviço de transporte, os *brokers* resolvem qual veículo de transporte deve levar o passageiro até o seu destino.
- No contexto de situações de emergência, os assinantes são as unidades móveis. As unidades móveis são veículos devidamente equipados com material de emergência e aparelhos possuindo pessoal qualificado e preparado para os mais diversos tipos de emergência. Elas se deslocam até a ocorrência e prestam o atendimento necessário dependendo da emergência. No serviço de transporte, os assinantes são os veículos que devem levar cada passageiro até seu destino.

Os *brokers* sempre decidem em conjunto, e de acordo com o modelo de comunicação por aptidão de assinante EMSS, quais os assinantes devem tratar cada evento.

4.1 Descrição Formal do Modelo EMSS

A comunicação entre os *brokers*, os assinantes e o publicador de eventos é especificada pelo modelo EMSS. Formalmente, essas entidades são denotadas conforme a Tabela 4.

Tabela 4 – Denotação das entidades do modelo EMSS.

Entidade	**Notação**	**Significado**
Brokers	$B = \{b_1, \ldots, b_q\}$	O conjunto dos *brokers* é denotado pelo *broker* de identificação 1 até o *broker* de identificação q
Assinantes	$A_i = \{a_{i1}, \ldots, a_{im}\}$	O conjunto dos assinantes do serviço i* (i. e., s_i) é denotado pelo assinante identificado por 1 até o assinante identificado por m
Publicador	P_i	Publicador do serviço i

*O serviço são as atividades necessárias e disponíveis na cidade para atender a um evento específico.

Fonte: Autoria Própria

O publicador do evento aciona o *broker* da região onde ocorreu o evento e este pode acionar os outros *brokers*, se necessário. O publicador informa o ponto de referência exato do evento. Os *brokers* notificam os assinantes necessários mais apropriados baseado no valor de aptidão (valor de *fitness*) normalizado entre 0 e 1. Para especificação do *fitness* faz-se necessário algumas definições que se encontram na Tabela 5.

Tabela 5 – Definições do modelo EMSS.

Definição	**Notação**	**Significado**
Pontos de referência	$R = \{r_1, \ldots, r_k\}$	O conjunto de todos os pontos de referência na cidade é denotado pelo primeiro ponto de referência até o $p-$ésimo ponto de referência
Serviços	$S = \{s_1, \ldots, s_n\}$	O conjunto dos serviços disponíveis na cidade é denotado pelo serviço identificado por 1 até o serviço n
Evento	e	O evento que acontece em qualquer região da cidade

Fonte: Autoria Própria

Um evento pode necessitar de vários assinantes para tratá-lo. Assim, o número de assinantes necessários é denotado por $N^e_{s_i} \in N$. Logo, a aptidão do assinante $a_{ij} \in A_i$ com relação a $r_k \in R$ é denotada por $fit(a_{ij}, k)$.

O modelo de eventos EMSS tem como objetivo escolher os assinantes disponíveis que apresentarem a melhor aptidão $fit(a_{ij}, k)$ do serviço $s_i \in S$ para tratar e. A disponibilidade do assinante é definida por D_{ij}, ou seja, a disponibilidade do assinante j pertencente ao serviço i. Os melhores assinantes de A_i devem ser selecionados de forma a maximizar uma função utilidade U_{s_i}, definida como a Equação 4.1:

$$\text{Maximizar} \quad U_{s_i} = \sum_{j=1}^{|A_{s_i}|} fit(a_{ij}, k) \cdot D_{ij} \tag{4.1}$$

$$\text{Sujeito a} \quad \sum_{j=1}^{|D_{ij}|} D_{ij} \leq N^e_{s_i}$$

onde,

$$D_{ij} = \begin{cases} 1, & \text{Se } a_{ij} \in A_i \text{ está disponível para tratar } e \\ 0, & \text{Caso contrário} \end{cases}$$

De acordo com a Equação 4.1, a função utilidade U_{s_i} é o somatório do produto do *fitness* de cada assinante j do serviço i em relação ao ponto de referência r_k, $(fit(a_{ij}, k))$, por sua disponibilidade, ou seja, 1 (disponível) ou 0 (indisponível). A maximização resultará em um conjunto de assinantes que pode ser inferior ou igual ao número de assinantes disponíveis necessários para tratar o evento, $N^e_{s_i}$. Como exemplo, imagine uma situação de emergência em que são necessárias duas ambulâncias, mas tem-se quatro ambulâncias no total e a segunda ambulância encontra-se indisponível. O ponto de referência do evento r_k é 7 e o serviço é identificado por 2. Logo, tem-se a função utilidade sendo composta

por: $U_{s_2} = fit(a_{21}, 7) \cdot 1 + fit(a_{22}, 7) \cdot 0 + fit(a_{23}, 7) \cdot 1 + fit(a_{24}, 7) \cdot 1$. Supondo que a primeira ambulância possui *fitness* igual a $0,9$, a segunda *fitness* com o valor 0, a terceira *fitness* em $0,8$ e a quarta *fitness* de $0,6$. Assim, se $N^e_{s_2} = 2$ e $U_{s_2} = 0,9 + 0,8 + 0,6$, então $U_{s_2} = 0,9 + 0,8$. Consequentemente, a primeira e a terceira ambulância foram selecionada para se deslocar até a emergência para atendê-la.

Desta forma, o subconjunto dos assinantes para o serviço $s_i \in S$ que representa os assinantes disponíveis para tratar o evento é dado por $\{a_{ij} \in A_i \mid D_{ij} = 1\}$. O principal desafio em resolver U_{s_i} para um dado serviço $s_i \in S$ é a determinação de $fit(a_{ij}, k)$ para todo $a_{ij} \in A_i$. O problema está no cálculo do *fitness* de cada assinante de um serviço específico que pode atender um evento reportado em certo ponto de referência. Assim, considerou-se as seguintes premissas:

i Enquanto o assinante está tratando o evento, ele está indisponível;

ii Quanto mais próximo o assinante estiver do ponto de referência, maior é o seu *fitness*;

iii Tráfego intenso significa mais tempo para um assinante alcançar um ponto de referência;

iv Quanto maior a jornada de trabalho de um assinante ou da equipe associada a ele, maior é o nível de fadiga do mesmo.

Considerando os fatores acima, definiu-se como métrica para o cálculo da aptidão de cada assinante disponível, os seguintes parâmetros:

a) O nível de tráfego entre o assinante a_{ij} localizado em (x_{ij}, y_{ij}) e o ponto de referência r_k localizado em (x_k, y_k), denotado por $Traf_{ij}$, o qual depende de:

   - A distância entre o assinante e o ponto de referência, d^k_{ij}, denotada pela Equação 4.2:

$$d^k_{ij} = \sqrt{(x_k - x_{ij})^2 + (y_k - y_{ij})^2} \tag{4.2}$$

   - A intensidade do tráfego, considerando a Norma Brasileira (NBR 5101:2018) que classifica o tráfego motorizado como leve, moderado e intenso, sendo descrita por c^k_{ij}, significando a intensidade de tráfego entre o assinante j do serviço i até o ponto de referência r_k.

   Logo, $Traf_{ij} = (d^k_{ij} \cdot c^k_{ij})$ e L_1 é um fator de normalização que depende da máxima distância entre dois pontos de referência quaisquer da área de cobertura do serviço na cidade.

b) O nível de fadiga do assinante denotado por $Fad_{ij} = Temp_{acum}/Temp_{max}$, onde $Temp_{acum}$ é o tempo acumulado para tratar os eventos e $Temp_{max}$ é o tempo máximo de atendimento observado durante um período específico, como por exemplo, 24 horas. O nível de fadiga é normalizado entre 0 e 1.

Assim, o *fitness* de cada assinante $a_{ij} \in A_i$ para cada ponto de referência $r_k \in R$ é calculado conforme a Equação 4.3:

$$fit(a_{ij}, k) = (1 - Traf_{ij}/L_1) \cdot (1 - Fad_{ij}/L_2) \tag{4.3}$$

onde, L_2 é um fator de escala do nível de fadiga associado ao assinante.

O cálculo do valor de aptidão proposto considera disponibilidade, distância, tráfego e nível de fadiga. No entanto, outros parâmetros podem ser considerados como, por exemplo, a classificação do assinante. A classificação mede o grau de satisfação que os utilizadores do serviço conferem ao assinante. Assim, um sistema de classificação com 6 estrelas foi adotado para isso. Logo, no cálculo do valor de aptidão proposto, esse parâmetro pode ser considerado como uma extensão. Quanto melhor a classificação do assinante, maior será o seu valor de *fitness*.

Desta forma, o *fitness* de cada assinante a_{ij} incluindo sua classificação é calculado conforme a Equação 4.4 abaixo:

$$fit(a_{ij}, k) = (1 - Traf_{ij}/L_1) \cdot (1 - Fad_{ij}/L_2) \cdot (1 - Class_j/L_3) \tag{4.4}$$

onde, L_3 é um fator de escala relacionado ao nível de classificação associado ao assinante. O nível de classificação de um assinante é denotado por $Class_j = (1 - C_{ij}/Q)$, em que $C_{ij} \in \{0, 1, \ldots, Q\}$ é o nível de classificação do assinante que é atualizado com base na experiência do publicador. O nível de classificação do assinante é normalizado entre 0 e 1.

4.2 O Middleware EMSS

O *middleware* EMSS implementa o modelo de evento EMSS. Como entidades básicas do *middleware* EMSS tem-se *brokers*, assinantes e o publicador que representa o sistema de eventos ou a aplicação que utiliza o *middleware* EMSS. Os *brokers* recebem os eventos enviados pelos publicadores e os assinantes emitindo subscrições consomem os eventos de acordo com o modelo de eventos implementado pelo *middleware*. Os *brokers* oferecem pontos de acesso para subscrições e para a publicação dos eventos, como por exemplo, para chamados ao atendimento de emergências. O publicador se comunica com os *brokers* para que o *middleware* possa publicar os eventos aos assinantes mais adequados de acordo com o modelo de eventos por aptidão de assinante EMSS.

4.2.1 Arquitetura do Serviço de Notificação de Eventos EMSS

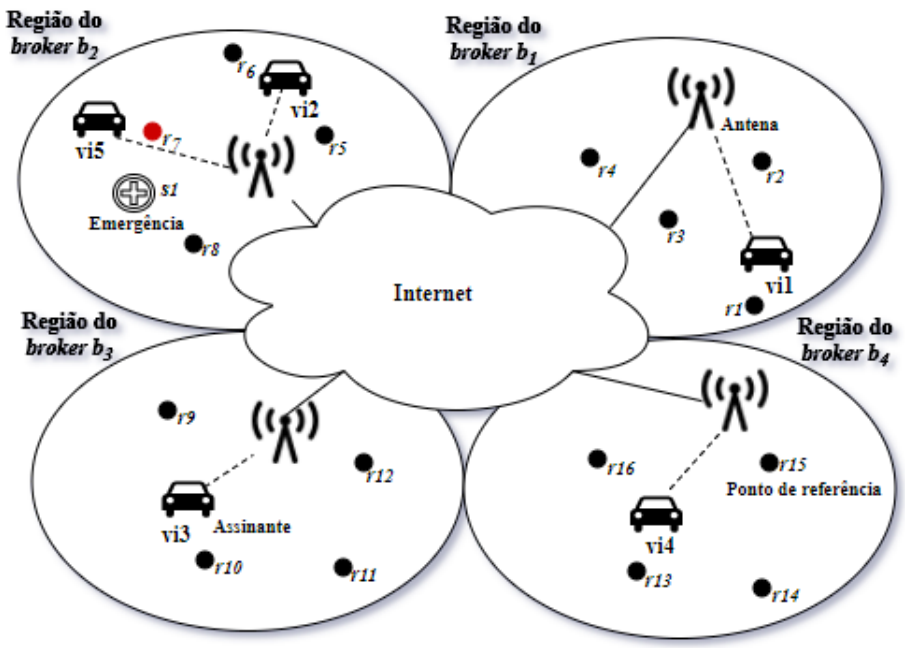
A arquitetura do serviço de notificação de eventos EMSS segue uma abordagem distribuída, na qual as atividades de coleta de subscrições e roteamento ficam espalhadas pela rede. Logo, o *middleware* EMSS é implementado como um conjunto de *brokers* distribuídos que cooperam entre si para entregar os eventos aos assinantes mais adequados. Os *brokers* são distribuídos (alocados) por regiões da cidade, um para cada região, em que cada *broker* cobre uma mesma área de abrangência/cobertura de uma antena da rede de telefonia móvel. Por consequência, a tomada de decisão é descentralizada, sendo que cada *broker* mantém uma parte do estado do sistema. A decisão descentralizada com alocação distribuída de *brokers* favorece a disponibilidade e a escalabilidade do sistema, pois, em princípio, não apresenta um ponto único falha e nenhum ponto crítico de sobrecarga (de processamento, armazenamento ou comunicação). O *broker* inicia o processo de tomada de decisão para todos os eventos que acontecem em sua região e a tomada de decisão baseia-se em regras de comunicação entre os *brokers* definidas pelo modelo de comunicação por aptidão de assinante EMSS. A tomada de decisão envolve apenas um subconjunto de *brokers*, ou seja, o *broker* da região em que ocorre o evento e os *brokers* das regiões com assinantes pertencentes ao serviço solicitado.

A Figura 18 ilustra a arquitetura do *middleware* EMSS, sendo que as seguintes premissas foram assumidas:

1. Uma rede de *Fogs* é criada para o envio de eventos.
2. Um *broker* pode executar em um servidor ou objeto inteligente (i. e., um *Fog node*) com capacidade de processamento e comunicação e é associado a uma antena de telefonia celular. Logo, o *broker* executa em um *Fog node* dentro da *Fog*.
3. A comunicação entre os *brokers* acontece através da Internet, assim cada *broker* pode se comunicar diretamente com outro.
4. Os assinantes móveis escolhem o *broker* mais próximo, ou seja, o *Fog node* associado ao *Fog* escolhido, para subscreverem-se de modo a receberem os eventos. A comunicação entre eles acontece através de rede sem fio (4G ou 5G).
5. O publicador (não mostrado na figura) se comunica com os *brokers* através da Internet.

A Figura 18 mostra uma cidade hipotética dividida em 4 regiões, cada uma contendo quatro pontos de referência. Os eventos são associados a pontos de referência. Cada região possui um único *broker* que é responsável pela gerência dos eventos daquela região e possui pontos de referência associados e gerenciados por ele. Na figura também estão representados os assinantes que são as unidades móveis e um evento de emergência acontecendo na região

Figura 18 – Arquitetura do *middleware* EMSS.



Fonte: Autoria Própria

do *broker* b_2. Uma infraestrutura fixa de comunicação juntamente com uma infraestrutura móvel é utilizada. Os *brokers* fazem parte da infraestrutura fixa e os assinantes podem ser considerados nós móveis e, por vezes nós fixos, se comportando como atuadores distribuídos que realizam alguma atividade quando são notificados. Essa arquitetura foi escolhida para atender aos requisitos de grandes sistemas distribuídos (i. e., escalabilidade, baixo atraso), como é o caso de sistemas que oferecem serviços em Cidades Inteligentes. Desta forma, tendo em vista que o *middleware* EMSS utiliza uma arquitetura distribuída, forma-se então uma rede de *brokers* que pode ser vista como uma rede overlay sobre uma rede física como a Internet.

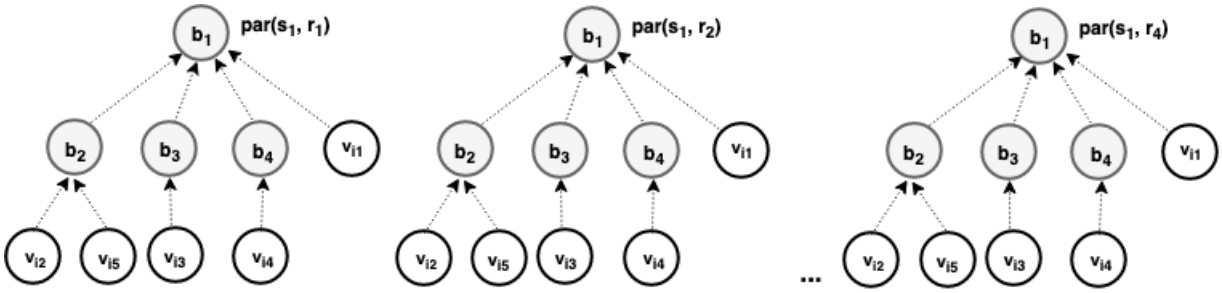
O padrão de comunicação utilizado foi o padrão de comunicação *Device-to-Gateway*. Cada *broker*, conforme o padrão, assume o papel de *gateway*. Com isso, comunica-se com os assinantes e com os outros *brokers*, toma as decisões de encaminhamento de eventos e transmite os dados coletados para um serviço de nuvem.

4.2.2 Redes Overlay

Os *brokers* se conectam entre si para trocar mensagens em relação a eventos formando assim redes overlay (ou de sobreposição) sobre a Internet, mais especificamente sobre a rede de *Fogs* criada. Em cada *broker*, uma rede overlay é criada para cada serviço $s_i \in S$ e ponto de referência $r_k \in R$, par (s_i, r_k). Cada rede criada segue uma topologia

hierárquica em formato de árvore. Essa organização lógica dos *brokers* foi escolhida com o propósito de minimizar o atraso no envio das mensagens entre as entidades do *middleware*. A Figura 19 mostra as redes overlay criadas no *broker* b_1 para o serviço s_1 e para os pontos de referência de sua região, tendo como base a cidade hipotética ilustrada na Figura 18. Na Figura 19, a terceira rede do par (s_1, r_3) não é mostrada.

Figura 19 – Redes overlay criadas no *broker* b_1.



Fonte: Autoria Própria

As redes overlay são utilizadas para a notificação de eventos aos assinantes mais apropriados de cada serviço necessário s_i e ponto de referência em que acontece o evento, r_k. Redes overlay devem ser construídas dinamicamente para que os eventos sejam roteados dos publicadores para todos os assinantes interessados (PIETZUCH; BACON, 2002). A função da rede overlay é fazer com que cada evento seja notificado ao subconjunto dos assinantes mais apropriados do serviço s_i em relação ao ponto de referência do evento, r_k. A topologia em árvore de cada rede é atualizada dinamicamente conforme os assinantes se movem dentro da cidade. Toda vez que um assinante troca de região ele virá folha da árvore do *broker* da região em que se encontra e é excluído da árvore do *broker* da região em que se encontrava anteriormente. O evento na árvore desce até o assinante selecionado ou nó folha para que ele possa tratá-lo. A raiz de cada árvore é o *broker* responsável por um ponto de referência específico de sua região, o qual posteriormente pode ser o ponto de referência do evento.

4.2.3 Subscrições para Eventos

A mensagem de subscrição ou inscrição para um evento é enviada por um assinante quando o mesmo entra na região de um *broker*. A subscrição é importante para que o assinante receba e trate os eventos de interesse, funcionando como um cadastro de assinantes interessados em um determinado evento. Desta forma, quando o *broker* recebe uma mensagem de subscrição (*subscribe message*) de um assinante, calcula os valores de *fitness* para cada ponto de referência da cidade com os valores recebidos da mensagem ou então para o ponto de referência específico do evento. A situação de trânsito de cada

região da cidade (obtida em tempo real) também é utilizada para o cálculo do *fitness*. Posteriormente, o *broker* associa o assinante a cada rede overlay correspondente ao serviço s_i e ao ponto de referência r_k pertencente ao *broker* e insere os valores de *fitness* calculados em relação ao par correspondente (s_i, r_k). O diagrama de estados da Figura 20 ilustra o fluxo de controle de estado para estado quando o *broker* de uma região recebe uma subscrição de um assinante.

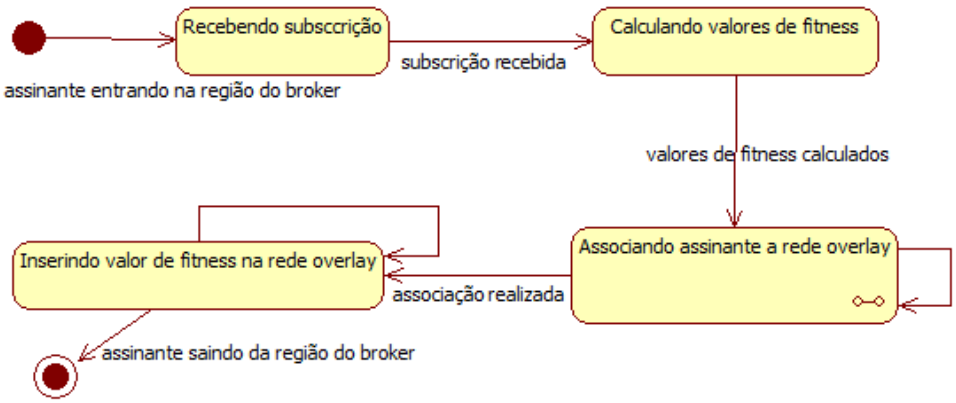


Figura 20 – Diagrama de Estados do *broker* representando os estados quando recebe uma mensagem de subscrição de um assinante.

O Algoritmo 1 mostra o cálculo do *fitness* em relação a um serviço específico s_i e ponto de referência r_k para um assinante. Para os serviços de emergência, o nível de classificação é menor do que 0 (zero). A complexidade do algoritmo em cada *broker* é *O(k*m)*, onde k é o número total de pontos de referência na cidade e m é a quantidade de assinantes do serviço s_i, pois para um ponto de referência r_k e um serviço s_i tem-se tempo constante (não depende do tamanho da entrada). *O(1)* é o tempo do algoritmo 1 para um ponto de referência. Como o algoritmo está sendo executado para k pontos e m assinantes, tem-se *O(1)*k*m*, logo a complexidade é de *O(k*m)*.

Com os valores de *fitness* inseridos em cada rede criada no *broker* pertencente ao par (s_i, r_k), um segundo passo é enviar aos outros *brokers* da cidade um único valor de *fitness* para cada par (s_i, r_k). Esse envio leva em conta os pontos de referência pertencentes a cada *broker*. Valores de *fitness* relacionados aos pontos de referência do *broker* em questão não são enviados aos outros *brokers*, pois cada *broker* mantém redes overlay de eventos pertencentes aos pontos de referência de sua região. Essas redes são chamadas de redes overlay de disseminação de eventos. Isso explica a abordagem da tomada de decisões dos *brokers* de forma distribuída, em que cada *broker* possui parte do estado do sistema de notificação de eventos (ou seja, os valores de *fitness*). Consequentemente, os *brokers* tomam decisões conjuntas sobre quais assinantes devem tratar o evento.

Algoritmo 1: Algoritmo para o cálculo do *fitness* em relação a um ponto de referência de um assinante para um serviço s_i.

Input: localização do assinante (x_2, y_2), disponibilidade do assinante ($disponibilidade$), trânsito ($transito$), maior distância entre todos os pontos da cidade (L_1), fator de escala para o nível de fadiga (L_2), fator de escala para o nível de avaliação (L_3), ponto de referência (x_1, y_1), nível de fadiga associado ao assinante ($nivelfadiga$), nível de classificação associado ao assinante ($nivelclassificacao$).

Output: *fitness* de um assinante para um ponto de referência específico da cidade.

```
begin
    distancia ← sqrt((sqr(x2 - x1) + sqr(y2 - y1));
    if (disponibilidade = "sim") then
        if (transito = "intenso") then
            coeficiente ← sorteio(0.0, 0.3));
        else
            if (transito = "moderado") then
                coeficiente ← sorteio(0.3, 0.6));
            else
                if (transito = "leve") then
                    coeficiente ← sorteio(0.6, 1.0));
                end
            end
        end
    else
        coeficiente ← 0;
    end
    if (nivelclassificação ≥ 0) then
        fitness[ponto] ← ( (1 - distancia/L1 * coeficiente) * (1 – nivelfadiga/L2) *
          (1 - nivelclassificacao/L3) );
    else
        fitness[ponto] ← ( (1 - distancia/L1 * coeficiente) * (1 – nivelfadiga/L2) );
    end
end
```

A tomada de decisão é iniciada pelo *broker* raiz da rede overlay que é acionado para tal. O *broker* é considerado raiz de uma rede overlay quando esta possui os valores de *fitness* de algum dos pontos de referência sob a sua região de abrangência. Quando o *broker* não for a raiz da rede ele é considerado como um *broker* intermediário. Quando o *broker* é a raiz da rede, os valores de *fitness* calculados, exceto os que fazem parte de sua região, são enviados aos outros *brokers* de acordo com os pontos de referência sob a responsabilidade de cada um. Para esse envio, é considerado o maior conjunto de valores

de *fitness* em relação aos pontos de referência r_k da região de abrangência do *broker* que receberá os valores. Como exemplo, se um *broker* tiver 20 assinantes subscritos, o mesmo só enviará ao *broker* raiz de outra rede overlay o maior conjunto de valores de *fitness* em relação a cada par (s_i, r_k), independente de qual assinante pertencer o maior valor encontrado. Para isso, além das redes overlay de disseminação de eventos, o *broker* raiz mantém estruturas secundárias com os valores de *fitness* dos assinantes sob sua área de cobertura em relação aos pontos de referência pertencentes aos outros *brokers* que estão distribuídos na cidade. Isso para que ele possa fazer a escolha do conjunto com os maiores valores de *fitness* a serem enviados a um *broker* específico. O Algoritmo 2 ilustra a descoberta do maior conjunto dos valores de *fitness* entre todas os assinantes subscritos em um *broker* raiz para os pontos de referência sob a abrangência de um *broker* pertencente a outra região da cidade. A complexidade do algoritmo é de *O(k*m)*, pois o comando *for* mais interno executa m vezes (i. e., o N do for) que é a quantidade de assinantes. Essa for é executado dentro de outro, que por sua vez, executa k vezes, que é a quantidade de pontos de referência na cidade.

Algoritmo 2: Algoritmo para descobrir o maior conjunto dos valores de *fitness* de todos os assinantes subscritos em um *broker* em relação aos pontos de referência pertencentes a um *broker* de outra região.

Input: número de assinantes (N), valores de *fitness* dos assinantes em relação aos pontos de referência pertencentes a um *broker* (*fitness*), número de pontos de referência ($nrpontosreferencia$).

Output: conjunto com os maiores valores de *fitness* dos assinantes em um *broker* para os pontos de referência específicos de outro *broker*.

```
begin
    for ponto ← 1 to nrpontosreferencia do
        maior ← 0;
        for i ← 1 to N do
            if (fitness[ponto] > maior) then
                maior ← fitness[ponto];
            end
        end
        conjunto[ponto] ← maior;
    end
end
```

Quando um *broker* intermediário recebe uma mensagem com os valores de *fitness*, atualiza as redes overlay para cada serviço s_i e ponto de referência r_k. Desta forma, o estado do sistema de eventos é atualizado no *broker*.

As atualizações são realizadas para garantir a eficácia computacional do sistema de eventos que utiliza o *middleware* EMSS. Ainda, a atualização de estado para um

determinado serviço e ponto de referência, par (s_i, r_k), pode ser realizada de acordo com uma das seguintes abordagens:

- **Abordagem proativa:** o valor do *fitness* é atualizado de tempos em tempos para que esteja sempre pronto quando necessário para receber e tratar o evento.
- **Abordagem reativa:** o valor do *fitness* é atualizado somente durante o momento em que o evento acontece.

A abordagem proativa permite ao *broker* tomar uma decisão imediata, uma vez que todos os dados necessários estão disponíveis e atualizados, enquanto na abordagem reativa o estado é atualizado no momento da tomada de decisão. Na abordagem reativa, o assinante envia a mensagem de subscrição somente quando um evento acontece na cidade. O *broker* da região onde ele se encontra solicita que ele se inscreva, caso ele tenha interesse em tratar o evento.

Ainda, os assinantes se movimentam pela cidade, atendem a eventos, podem apresentar problemas ficando inoperantes, entre outros. Assim, o *fitness* de um assinante para um ponto de referência e serviço precisa ser atualizado constantemente, ou seja, o assinante precisa manter o *broker* da região em que se encontra atualizado periodicamente sobre seu estado. Para isso, o assinante envia mensagens de *bind* periodicamente para o *broker* que gerencia a região da cidade onde se encontra. Isso na abordagem proativa. A mensagem *bind* só pode ser enviada depois de uma mensagem *subscribe* ter sido recebida pelo assinante. Na abordagem reativa, uma única mensagem de *bind* é utilizada por um assinante quando ele finaliza o tratamento de um evento. Isso para que, se necessário, o *broker* possa acioná-lo novamente.

Desta forma, com as informações recebidas pela mensagem *bind*, o *broker* recalcula os valores de *fitness* do assinante em relação aos pontos de referência da cidade. Posteriormente, atualiza a topologia de cada rede overlay para cada serviço e ponto de referência, par (s_i, r_k), pertencente ao *broker* que recebeu *bind*, bem como as estruturas secundárias de armazenamento dos valores de *fitness* dos outros pontos de referência da cidade. O Algoritmo 3 descreve o que acontece quando o *broker* recebe uma mensagem *bind* de um assinante e recalcula os valores de *fitness* para cada par (s_i, r_k).

A complexidade do algoritmo em cada *broker*, sendo k é o número total de pontos de referência na cidade e m é a quantidade de assinantes do serviço s_i é *O(k, m)*, pois para um ponto de referência r_k e um serviço s_i tem-se um tempo constante. *O(1)* é o tempo do algoritmo 3 para um ponto de referência; e, como o algoritmo está sendo executado para k pontos e m assinantes, tem-se *O(1)*k*m*, que é igual a *O(k*m)*.

Algoritmo 3: Algoritmo para atualização de estado de um assinante.

Input: localização do assinante (x_2, y_2), disponibilidade do assinante (*disponibilidade*), serviço (s_i), ponto de referência (x_1, y_1), nível de fadiga associado ao assinante ($nivelfadiga$), nível de classificação associado ao assinante ($nivelclassificacao$) .

Output: atualização do *fitness* de um assinante para um ponto de referência específico da cidade.

```
begin
    ponto = obtemponto(x1, y1);
    if ( serviço = si) and (ponto = rk) then
        if ( disponibilidade = "sim") then
            if (nívelclassificação ≥ 0) then
                fitness[ponto] = calculafitness(si, x2, y2, disponibilidade,
                 nivelfadiga[assinante], nivelavaliacao[assinante]);
            else
                fitness[ponto] = calculafitness(si, x2, y2, disponibilidade,
                 nivelfadiga[assinante]);
            end
        else
            fitness[ponto] = 0;
        end
    end
    atualizaredeoverlay(fitness[ponto]);
end
```

Com as topologias das redes overlay atualizadas, o estado do assinante se modifica. O envio dos valores de *fitness* calculados para cada par (s_i, r_k) pertencente aos *brokers* das outras regiões da cidade acontece. Por consequinte, as redes overlay dos *brokers* das outras regiões da cidade são atualizadas. O diagrama de estados da Figura 21 ilustra o fluxo de controle de estado para estado quando o *broker* de uma região recebe um mensagem de *bind* de um assinante.

Quando o assinante deseja sair da região em que se encontra, deve enviar uma mensagem de cancelamento da subscrição (*unsubscribe message*). Com isso, o *broker* que recebe a mensagem exclui o assinante atualizando as redes overlay e, se necessário, envia novos valores de *fitness* aos *brokers* das outras regiões da cidade. O diagrama de sequência da Figura 22 ilustra a sequência de atividades que acontecem durante a interação entre as entidades do *middleware* EMSS em função da troca das mensagens *subscribe*, *bind*, *unsubscribe* ou *ack-notification* (mensagem de confirmação de acionamento de assinante para tratar o evento). Embora a mensagem *ack-notification* não aparece no diagrama, a sequência de atividades depois do recebimento dela pelo *broker* é a mesma das mensagens

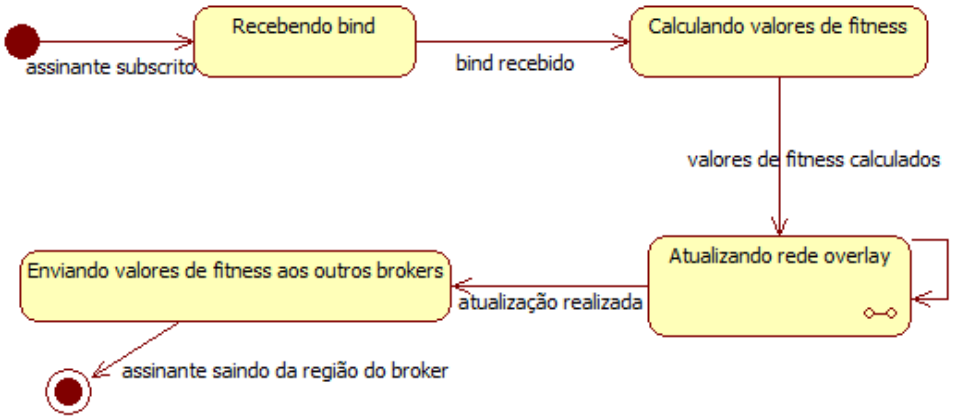


Figura 21 – Diagrama de Estados do *broker* representando os estados quando recebe uma mensagem de *bind* de um assinante.

subscribe e *bind*.

Finalmente, na ausência de assinantes disponíveis para responder a um chamado, o publicador tenta resolver pedidos pendentes de assinantes por mensagens enviadas novamente a cada um minuto durante uma hora, até que o chamado seja cumprido ou finalmente cancelado. Se durante o reenvio de um pedido não atendido, um novo pedido for gerado, o mesmo é inserido em uma fila para posterior envio.

4.2.4 Notificação de Eventos

Uma ocorrência de um evento implica na notificação do evento para os assinantes do serviço necessário/requerido. Para a notificação do evento, o *broker* que recebe o aviso de que um evento deve ser tratado, por meio de uma mensagem de notificação do sistema de eventos (*notification-system message*), se comunica com outros *brokers* para escolher o assinante que será notificado do evento (ou seja, o assinante que tratará o evento). O *broker* que recebe o aviso é o *broker* raiz da rede overlay de disseminação de eventos ou também pode ser chamado de *broker* publicador.

4.2.4.1 Notificação na Abordagem Proativa

Se a abordagem proativa for utilizada para a notificação de eventos, quando o *broker* publicador receber uma mensagem *notification-system*, ele selecionará a rede overlay que corresponde ao par (s_i, r_k). Em seguida, escolherá os assinantes que receberão o evento. Se o assinante estiver na mesma região do *broker* publicador e for escolhido, uma mensagem de notificação (*notification message*), será enviada diretamente para este assinante. Caso contrário, se outros *brokers* (os *brokers* intermediários) precisarem ser ativados, uma mensagem de publicação (*publish message*) será enviada aos *brokers* intermediários que, então, notificarão alguns de seus assinantes por meio de mensagens *notification*. O Algoritmo 4 ilustra quando uma mensagem de *notification-system* é recebida

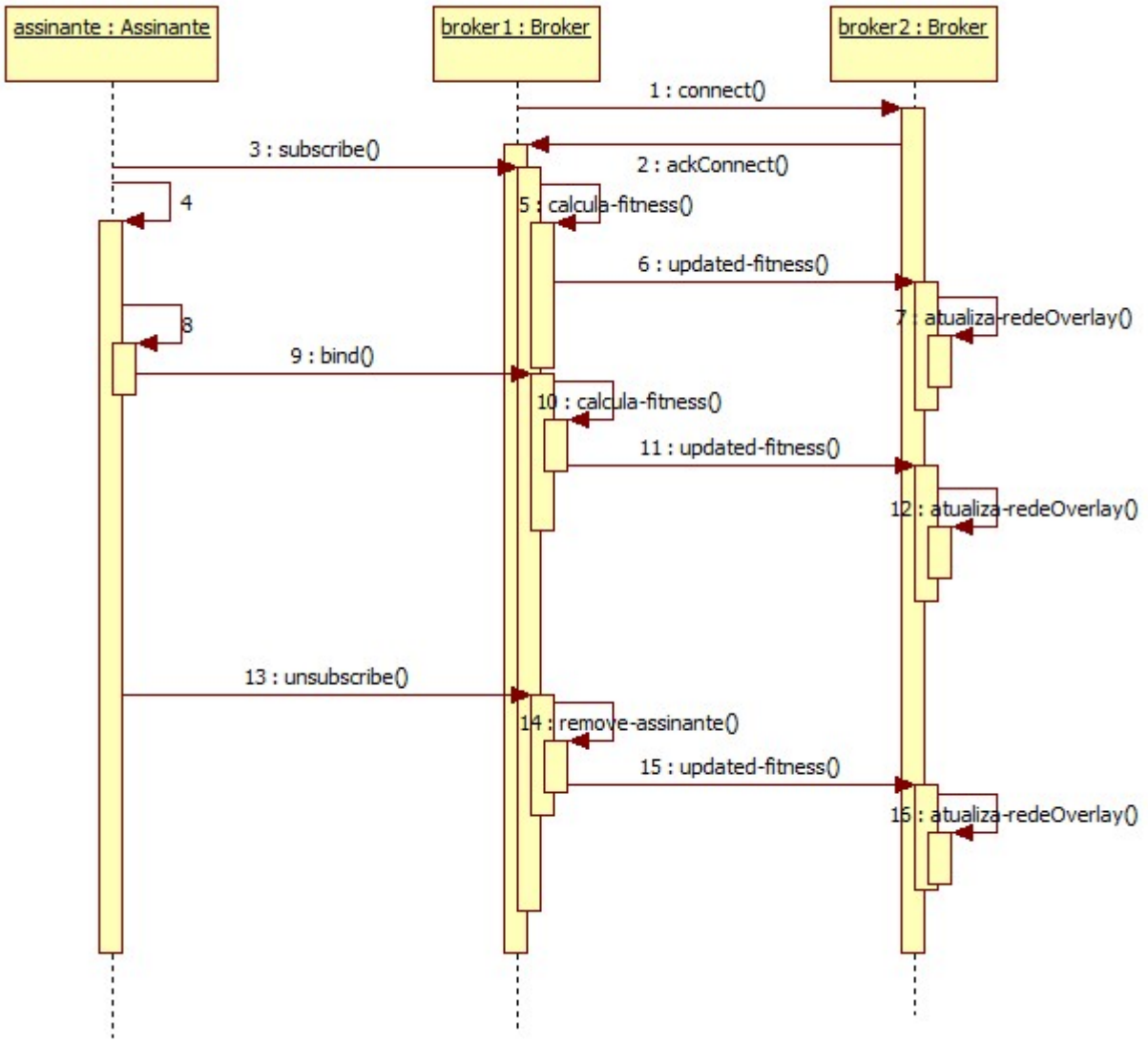


Figura 22 – Diagrama de sequência relacionado a troca das mensagens *subscribe*, *bind* e *unsubscribe*.

pelo *broker* publicador. O maior valor de *fitness* em relação ao ponto de referência em que correu a situação de emergência, r_k, é encontrado. A identificação do *broker* intermediário ou do assinante a que pertence o maior valor de *fitness* é armazenada. Por consequente, o assinante ou o *broker* intermediário recebem as mensagens *notification* ou *publish* com o evento, respectivamente. A complexidade do algoritmo 4 em cada *broker* publicador é $O(x, max(m^2, m + k/b))$, pois o laço mais externo (i. e., *repeat*) executa no máximo x vezes, ou seja, N, e o laço interno (*for*) executa $m + k/b$, onde m é o número de assinantes somando ao número de *brokers* que é dado por k igual ao número total de pontos de referência na cidade e b igual ao número de pontos de referência por *broker*. Entretanto, a complexidade do laço mais externo depende do *for* interno ou da função *publish* que é

$O(m^2)$.

Algoritmo 4: Algoritmo para notificação de eventos pelo *broker* publicador.

Input: número de assinantes necessários para o evento (N), serviço (s_i), ponto de referência do evento (r_k), *fitness* de cada assinante em relação aos pontos de referência de um serviço s ($fitness$).

Output: envio de evento para os assinantes.

```
begin
    cont ← 0;
    existechange ← V;
    repeat
        maior ← -1;
        if (serviço = s_i) and and (ponto = r_k) then
            for i ← 1 to nrfilhosarvore do
                if (fitness[ponto] > maior) then
                    maior ← fitness[ponto];
                    assinanteselecionado ← i;
                end
            end
            if (assinanteselecionado > quantidadebrokers) then
                ack = notification(assinanteselecionado, s_i, r_k);
            else
                ack = publish(assinanteselecionado, s_i, r_k);
            end
            if (ack = V) then
                cont ← cont + 1;
            else
                existechance ← F;
            end
            if cont = N then
                existechance ← F;
            end
        end
    until ( (cont < N) e (existechance = V) );
    return cont;
end
```

Quando um *broker* intermediário recebe uma mensagem de *publish*, ele precisa acionar algum de seus assinantes, o mais apropriado, para receber o evento. O serviço s_i solicitado e o ponto de referência r_k em que o evento ocorreu precisa ser informado a esse *broker* para que ele possa selecionar algum dos seus assinantes. Desta forma, ele notificará o assinante com as informações necessárias sobre o evento. O Algoritmo 5 ilustra o código

de *publish*, o qual é executado por um *broker* intermediário. A complexidade do algoritmo, executado em cada *broker* intermediário é de $O(m^2)$, pois o laço de repetição externo pode executar até m que é o número de assinantes disponíveis (no algoritmo N), bem como o laço interno.

Algoritmo 5: Algoritmo para notificação de eventos a assinante pertencente a um *broker* intermediário.

Input: número de assinantes (N), *fitness* de cada assinante em relação ao ponto de referência do evento ($fitness$).

Output: notificação do evento para o assinante que apresentar o maior *fitness*, de todas os assinantes de um *broker* intermediário, para o ponto de referência em que o evento ocorreu.

```
begin
    existechange ← V;
    repeat
        maior ← -1;
        if (serviço = s_i) and and (ponto = r_k) then
            for i ← 1 to N do
                if (fitness[ponto] > maior) then
                    maior ← fitness[ponto];
                    assinanteselecionado ← i;
                end
            end
            ack = notification(assinanteselecionado, s_i, r_k);
            if (ack = V) then
                existechance ← F;
            end
        end
    until (existechance = V );
    return existechance;
end
```

As mensagens *publish* e *notification* são confirmadas. No entanto, se o assinante deixar a região do *broker* publicador ou do *broker* intermediário durante a mensagem *notification*, outro assinante será notificado. Antes da operação *notification*, o *broker* verifica se o assinante se encontra na região. Em caso afirmativo, envia a mensagem; caso contrário, outro assinante é selecionado e notificado para atendimento ao evento. O assinante quando muda de região envia uma mensagem específica para isso (mensagem *unsubscribe*). O *broker*, como consequência, detecta sua saída e atualiza a topologia das redes overlay de disseminação de eventos. No caso do Algoritmo 5, essa saída é demonstrada pelo retorno da chamada à operação *notification* ($ack = F$). O diagrama de estados da Figura 23 representa o fluxo de estados de forma simplificada quando um evento é recebido pelo

broker publicador até este enviar uma resposta de confirmação ao publicador do evento.

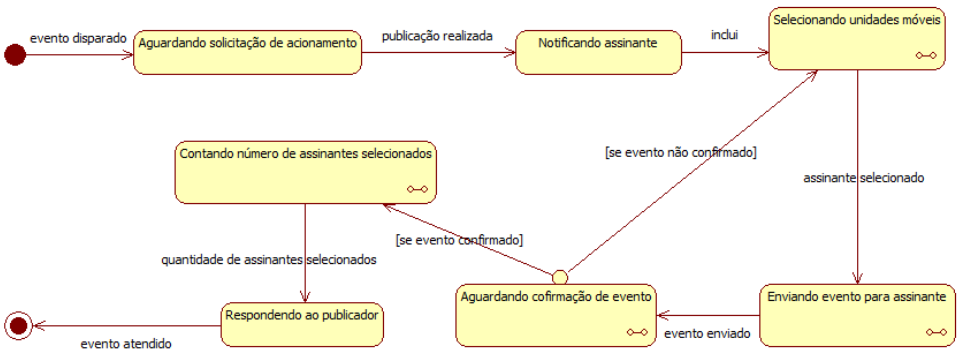


Figura 23 – Diagrama de Estados referente a evento recebido por *broker* publicador

A confirmação da mensagem *publish* contém um *flag* informando se algum dos assinantes do *broker* intermediário foi acionado ou não. Pode acontecer que o assinante deixou a região do *broker* enquanto a mensagem de *notification* estava sendo entregue ao mesmo. O único caso em que o *flag* pode ser falso, considerando que exista a mobilidade dos assinantes, é quando em uma última tentativa de notificação, um último assinante saia da região de abrangência do *broker* intermediário. Por fim, o *broker* publicador responderá ao sistema de eventos com uma mensagem de confirmação (*ack-notification-system message*) após a confirmação de cada mensagem *notification* ou de *publish*. Assim, a mobilidade dos assinantes é tratada pelo *middleware* EMSS. O diagrama de sequência da Figura 24, ilustra a sequência de atividades que ocorre durante a interação entre as entidades do *middleware* EMSS em função da troca de mensagens entre as entidades quando uma notificação de eventos acontece.

Como exemplo de notificação de eventos, a Figura 25 será utilizada. Na figura, a cidade foi dividia em quatro regiões, cada uma com quatro pontos de referência em que cada *broker* ficou responsável pelos pontos de referência presentes em sua região. O serviço escolhido foi bombeiro e uma situação de emergência para este serviço s_1 ocorre na região do broker b_2. A situação de emergência é associada ao ponto de referência r_7 que passa a ser o ponto de referência da emergência, r_k. O atendente envia uma mensagem de *notification-system message* para b_2 com a necessidade de três unidades móveis, N_1^e=3, para o serviço s_1. A figura mostra a rede overlay criada em b_2 para o par (s_1, r_k) com valores de *fitness* hipotéticos para cada entidade. O *broker* b_2 precisa dos valores de *fitness* das unidades móveis de sua região, bem como os valores de *fitness* de cada *broker* intermediário para fazer a escolha de quais deles devem ser notificados. Os valores de *fitness* em cinza não são visualizados pelo *broker* raiz, mas enviados a ele. O *broker* b_2 é a raiz da árvore (nível 0) porque é o responsável pela região onde a situação de emergência foi relatada. Os *brokers* intermediários b_1, b_3 e b_4, e as unidades móveis de b_2 (v_{12} e v_{15}) estão no nível 1. No nível 2 estão as unidades móveis v_{11}, v_{13} e v_{14} gerenciadas pelos *brokers* intermediários b_1, b_3 e b_4, respectivamente. Supõe-se que a unidade móvel v_{12} esteja indisponível e que

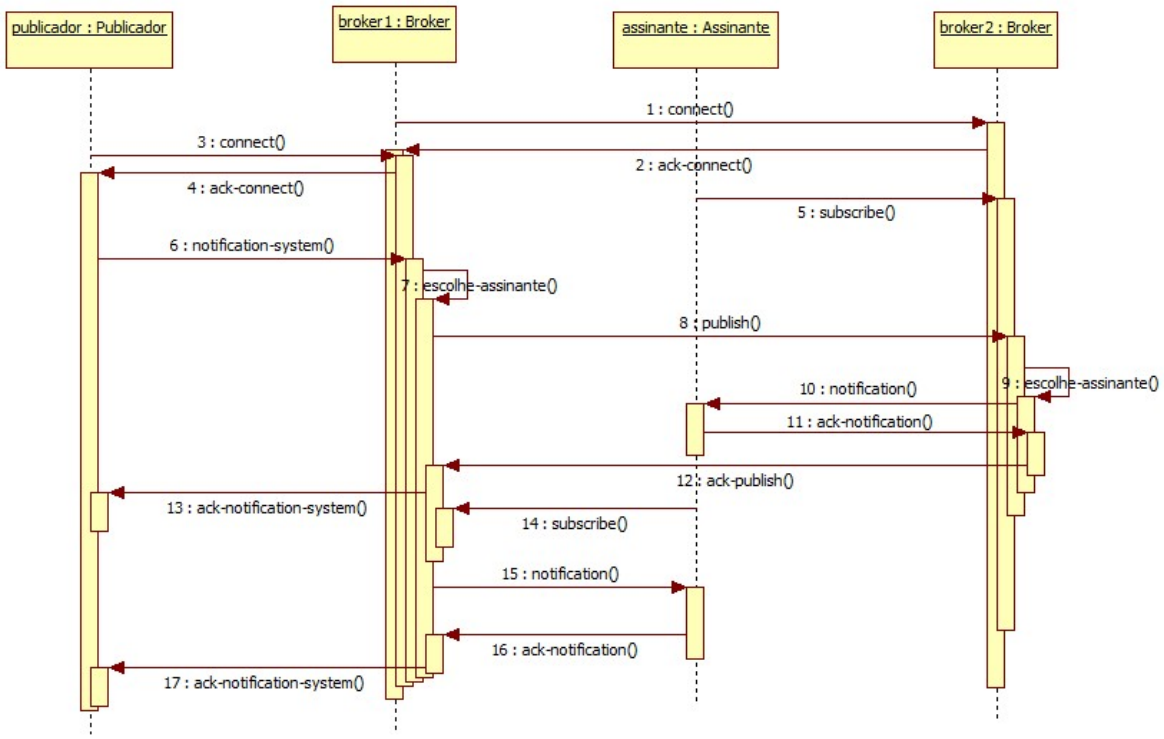


Figura 24 – Diagrama de sequência relacionada a notificação de eventos (abordagem proativa).

as condições de tráfego foram consideradas moderadas. Os valores de *fitness* hipotéticos considerados foram $fit_(v_{11})^7 = 0,6$, $fit_(v_{12})^7 = 0,0$, $fit_(v_{13})^7 = 0,7$, $fit_(v_{14})^7 = 0,5$ e $fit_(v_{15})^7 = 0,8$. Desta forma, as unidades móveis selecionadas serão v_{11}, v_{13} e v_{15}. Assim, o *broker* publicador b_2 enviará uma mensagem *notification* para a unidade móvel v_{15} e enviará mensagens *publish* para os *brokers* intermediários b_1 e b_3 para ativarem suas unidades móveis v_{11} e v_{13}, respectivamente. Cada *broker* intermediário que recebe *publish* envia uma mensagem *notification* para a unidade móvel que ele seleciona para tratar o evento.

4.2.4.2 Notificação na Abordagem Reativa

Se a abordagem reativa for utilizada para a notificação de eventos, quando o *broker* publicador receber o aviso do evento:

a: Solicitará aos assinantes de sua região que se subscrevam por meio de uma mensagem *invite-subscribe*.

b: Receberá as mensagens de subscrição e, como consequência, calculará o valor do *fitness* de cada assinante para o ponto de referência específico do evento.

c: Associará na rede overlay criada para o serviço e ponto de referência do evento, par (s_i, r_k), os assinantes subscritos adicionando os valores de *fitness* calculados.

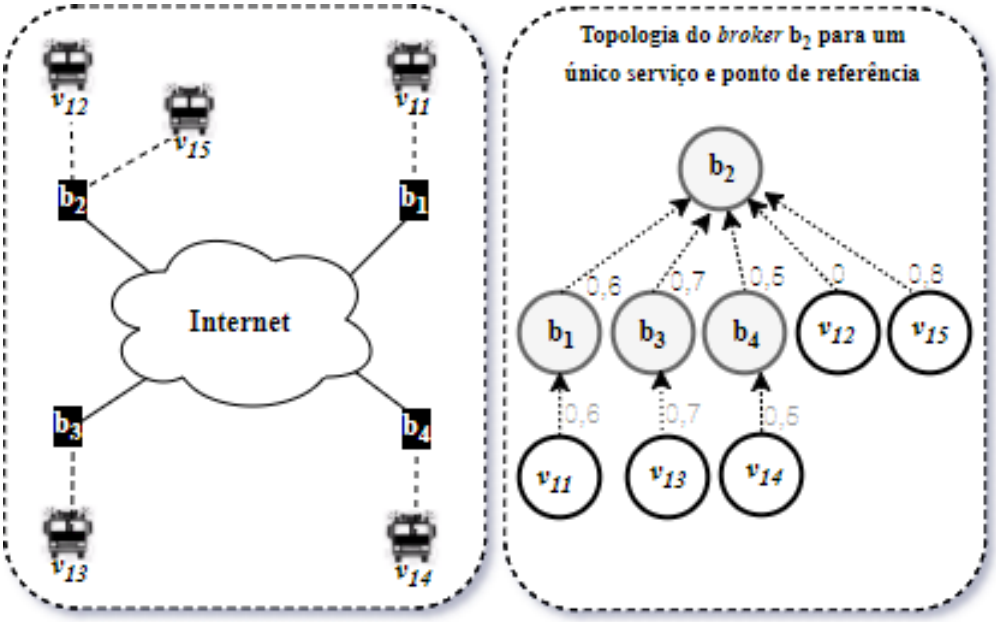


Figura 25 – Exemplo de Rede Overlay no broker b_2.

d: Requisitará a cada *broker* intermediário que forneça o maior valor de *fitness* dos assinantes que estão em suas regiões (mensagem *invite-updated-fitness*). Como consequência, cada *broker* intermediário:

- requisitará aos assinantes de sua região que se subscrevem para o evento (mensagem *invite-subscribe*), caso possam tratá-lo;
- calculará o valor de *fitness* para o ponto específico do evento para cada assinante;
- enviará o maior valor de *fitness* relacionado ao ponto de referência do evento por meio de uma mensagem *updated-fitness*.

e: Adicionará os valores de *fitness* recebidos de cada *broker* intermediário na rede overlay criada para o serviço e ponto de referência do evento.

f: Decidirá acionar o(s) assinante(s) mais apropriado(s) de sua região, se existir, ou o(s) assinante(s) mais apropriado(s) dos *brokers* intermediários, conforme o número dos assinantes requisitados pelo publicador. Se o *broker* intermediário precisar notificar algum de seus assinantes confirmará com o *broker* publicador o acionamento realizado.

g: Confirma o acionamento de unidades móveis ao publicador.

Os Algoritmos 4 e 5 usados para a notificação de eventos pelo *broker* publicador e para a notificação de eventos ao assinante pertencente a um *broker* intermediário, respectivamente, são utilizados na abordagem reativa. A única mudança se dá em relação aos pontos de referência que passam a ser um único ponto de referência, o ponto de referência do evento, r_k. O diagrama da Figura 26 ilustra a sequência de atividades, na

abordagem reativa, que acontecem em função da troca de mensagens entre as entidades do *middleware* EMSS quando o aviso de uma notificação de evento chega ao *broker* publicador. A conexão entre o *broker* publicador e o assinante foi suprimida.

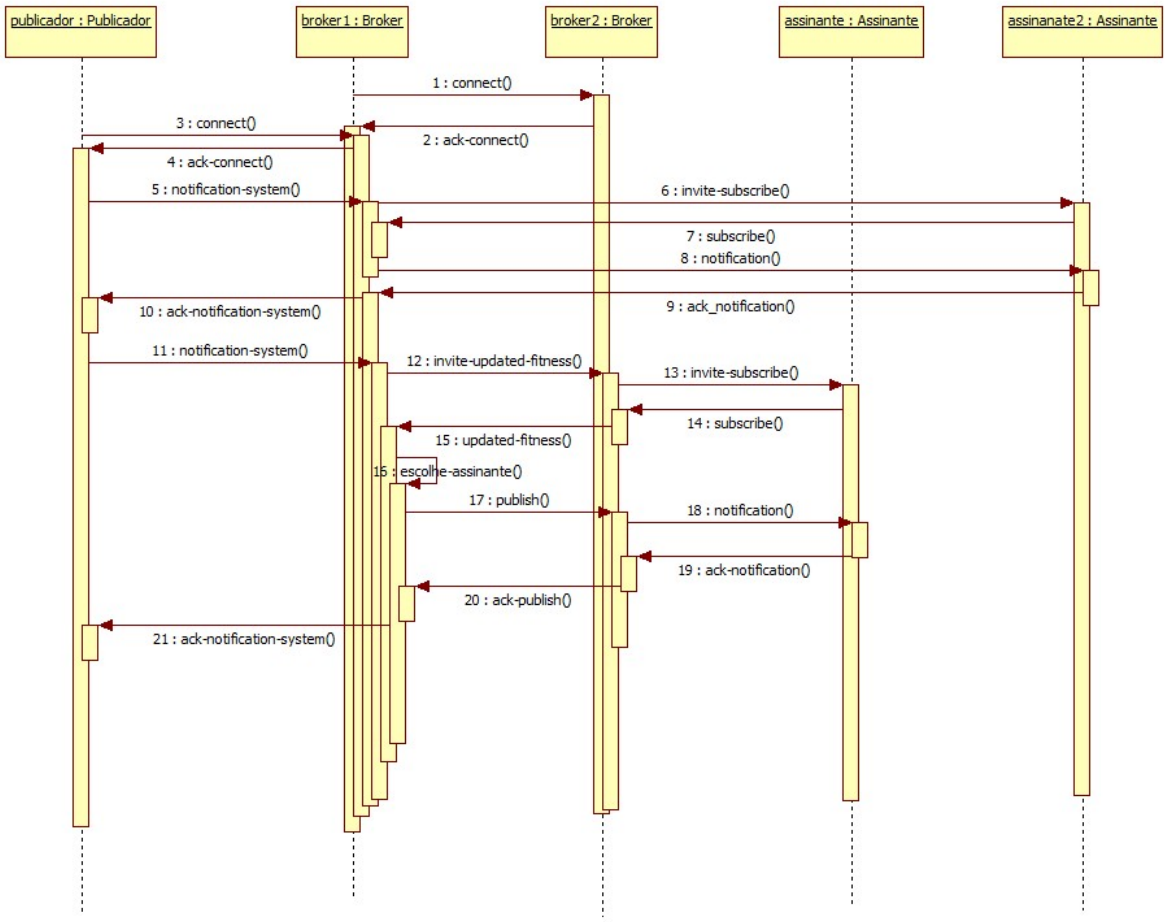


Figura 26 – Diagrama de sequência relacionada a notificação de eventos (abordagem reativa).

Por fim, o que muda de uma abordagem para outra é que na abordagem reativa, o *broker* que recebe a notificação do sistema não tem um estado pronto com os valores de *fitness* de seus assinantes e precisa os solicitar. Quando ele solicita, obtêm os valores de *fitness* somente do ponto de referência do evento. A decisão é tomada em reação a ocorrência do evento.

4.2.5 Mensagens das Entidades do Middleware EMSS

As entidades do *middleware* EMSS trocam mensagens entre si para desempenhar seus pápeis e funções. O diagrama de classes da Figura 27 descreve as entidades do *middleware* e o relacionamento entre elas.

As mensagens trocadas entre o publicador e o *broker* publicador são:

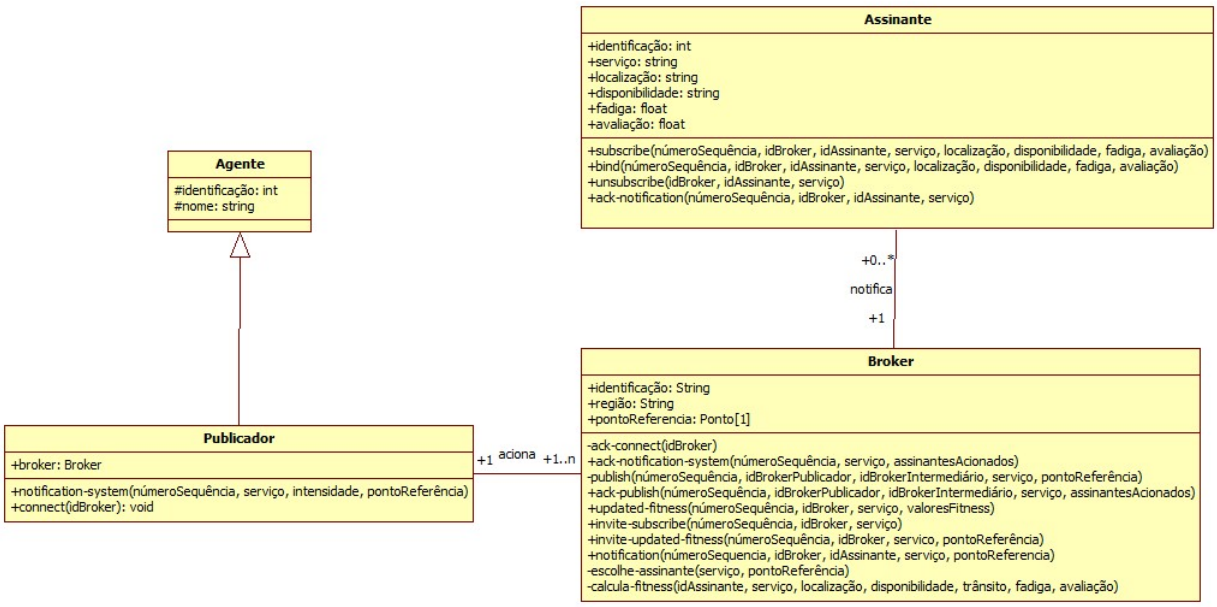


Figura 27 – Diagrama de classes do *middleware* EMSS.

1. ***notification-system***: mensagem enviada por um publicador para um *broker* avisando sobre a ocorrência de um evento na região pertencente ao *broker*. Indica o número de assinantes necessários para atender o evento. Os campos dessa mensagem são: número de sequência, serviço, ponto de referência do evento e número de assinantes necessários para o evento.

2. ***ack notification-system***: mensagem de confirmação de uma mensagem *notification-system* enviada por *broker* publicador em resposta a uma mensagem *notification-system*. A cada assinante acionado, o publicador recebe essa mensagem de confirmação do *broker* publicador. Desta forma, o publicador controla se o pedido é cumprido. Os campos dessa mensagem são: número de sequência, serviço, número de assinantes acionados (1 - um acionado ou 0 - nenhum acionado).

A mensagem enviada por um *broker* para um assinante é a mensagem de:

1. ***notification:*** mensagem que notifica o assinante sobre o evento que necessita ser tratado. Os campos dessa mensagem são: número de sequência, identificação do *broker* que envia a mensagem, identificação do assinante que irá atender o evento, serviço e ponto de referência do evento.

2. ***invite-subscribe:*** mensagem enviada por um *broker* publicador solicitando aos assinantes de sua região que se inscrevam para tratar do evento que acabou de acontecer. Os campos dessa mensagem são: número de sequência, identificação do *broker* que envia a mensagem e serviço. Mensagem utilizada somente na abordagem reativa.

As mensagens enviadas por um assinante para um *broker* são:

1. ***subscribe:*** mensagem de inscrição (subscrição) para um serviço. Os campos dessa mensagem são: número de sequência, identificação do *broker* da região em que o assinante se encontra, identificação do assinante que está enviando *subscribe*, coordenadas geográficas em que o mesmo se encontra, serviço, disponibilidade, nível de fadiga associado ao assinante e nível de avaliação associado ao assinante, se existir.

2. ***bind:*** mensagem de vínculo emitida de forma periódica informando que o assinante mantém-se na região de cobertura do *broker*. Enviada somente se a mensagem *subscribe* for recebida pelo assinante. Os campos dessa mensagem são: número de sequência, identificação do *broker*, identificação do assinante, coordenadas geográficas em que o assinante se encontra, serviço, disponibilidade, nível de fadiga e nível de avaliação, se existir, associados ao assinante.

3. ***unsubscribe:*** mensagem que sinaliza que o assinante está cancelando sua subscrição para um serviço específico. Os campos dessa mensagem são: identificação do *broker* que o assinante está deixando a região, identificação do assinante e serviço.

4. ***ack-notification:*** mensagem de confirmação de acionamento em resposta a uma mensagem *notification*. Os campos dessa mensagem são: número de sequência, identificação do *broker* que enviou *notification*, identificação do assinante acionado e serviço.

As mensagens enviadas entre *brokers* são:

1. ***publish:*** mensagem enviada por um *broker* publicador para um *broker* intermediário solicitando que ele acione algum de seus assinantes para atendimento ao evento. Os campos dessa mensagem são: número de sequência, identificação do *broker* publicador, identificação do *broker* intermediário, serviço e ponto de referência do evento.

2. ***ack-publish:*** mensagem que confirma o acionamento de assinante. Esta mensagem é enviada por um *broker* intermediário a um *broker* publicador em resposta a uma mensagem *publish*. Os campos dessa mensagem são: número de sequência, identificação do *broker* que emitiu *publish*, identificação do *broker* que está confirmando a mensagem de *publish*, serviço e número de assinantes acionados (1 - um acionado ou 0 - nenhum acionado).

3. ***invite-updated-fitness:*** mensagem enviada por um *broker* publicador aos *brokers* intermediários solicitando valores de *fitness* atualizados do ponto de referência do evento. Os *brokers* intermediários enviam mensagem de *invite-subscribe* aos assinantes de suas regiões quando recebem essa mensagem. Os campos dessa mensagem são:

número de sequência, identificação do *broker* que está solicitando está mensagem, serviço e ponto de referência do evento. Mensagem utilizada somente na abordagem reativa.

4. ***updated-fitness:*** mensagem enviada por um *broker* intermediário a um *broker* publicador contendo os valores de *fitness* atualizados dos assinantes de sua região. Um conjunto com os valores de *fitness* dos assinantes que estão na região do *broker* intermediário com o maior valor de cada ponto de referência do *broker* publicador para cada serviço. Isso na abordagem proativa. Na abordagem reativa, somente o maior valor de *fitness* em relação ao ponto de referência associado ao evento é enviado pelo *broker* em resposta a uma mensagem *invite-updated-fitness*. Os campos dessa mensagem são: número de sequência, identificação do *broker* intermediário, serviço e valores de *fitness* atualizados.

Salienta-se que uma mensagem *update-fitness* é enviada toda a vez que uma mensagem *subscribe*, *bind*, *unsubscribe* ou *ack-notification* é recebida por um *broker*. Essas são as mensagens utilizadas para comunicação entre as entidades do *middleware* EMSS.

5 Avaliação do Modelo de Eventos EMSS

O modelo de eventos EMSS foi avaliado para testar a hipótese de pesquisa planejada para a condução deste trabalho e alcançar o objetivo proposto. Assim, para afirmar ou refutar a hipótese de pesquisa e atingir o objetivo, um método de pesquisa foi adotado.

5.1 Método de Pesquisa

Um método é um conjunto de atividades sistemáticas que permite alcançar um objetivo, traçando o caminho a ser seguido, detectando erros e ajudando nas decisões do cientista (MARCONI; LAKATOS, 2004). Uma variedade de métodos pode ser aplicado em qualquer problema de pesquisa e vários métodos podem ser combinados para um amplo entendimento do problema. A escolha dos métodos depende da postura teórica dos pesquisadores, do acesso aos recursos e de como alinhar os métodos com as questões de pesquisa que têm sido colocadas (EASTERBROOK et al., 2008). O projeto de pesquisa é o processo de selecionar um método de pesquisa para um problema específico. Desta forma, para o desenvolvimento deste trabalho o método experimentação foi escolhido.

Experimentação e coleta de dados são as ferramentas da ciência para a validação de teorias (ZELKOWITZ; WALLACE, 1997). Uma pré-condição para conduzir um experimento é ter uma hipótese clara (EASTERBROOK et al., 2008). A hipótese e a teoria da qual ela vem guia todos os passos do projeto experimental, podendo ser realizada em um ambiente controlado como um laboratório, por exemplo. Neste caso, a experimentação é denominada de experimento controlado. Um dos experimentos controlados é a simulação (ZELKOWITZ; WALLACE, 1997). Na simulação, o pesquisador executa o produto com dados artificiais frequentemente em um modelo do ambiente real. A limitação da simulação está em quão bem o modelo corresponde ao ambiente real. Assim, quanto mais próximo o modelo estiver do ambiente real melhor.

Desta forma, a simulação foi utilizada neste trabalho para testar a hipótese de pesquisa e alcançar o objetivo proposto devido às limitações na utilização de outros métodos controlados, como por exemplo, o experimento replicado. Assim, para a execução da simulação realizou-se diferentes atividades, as quais foram adaptadas de (CRESWELL; CRESWELL, 2017):

1. **Definição:** nesta fase, o contexto do trabalho foi apresentado, o problema e os objetivos do trabalho foram definidos. Como resultado, a direção geral do experimento foi delineada. Essa fase pode ser encontrada na Seção 1 do trabalho.

2. **Planejamento do experimento:** envolveu a formulação da hipótese (seção 1), a seleção das variáveis (i. e., número de *brokers*, unidades móveis e assinantes), o projeto do experimento (i. e., área da cidade a ser simulada, modelo de distribuição dos dados), a preparação conceitual da instrumentação (i. e., as métricas de desempenho adotadas) e considerações sobre a validade do experimento. O experimento foi formulado por meio da hipótese delineada. Como resultado dessa fase, o experimento foi elaborado e ficou pronto para execução. O planejamento encontra-se na Seção 5.2. Nesta seção, são apresentados:

   a) o simulador utilizado (subseção 5.2);
   b) os estudos de caso simulados (subseção 5.2.1);
   c) a definição da área da cidade simulada (subseção 5.2.2);
   d) os chamados para tratar os eventos que aconteceram na cidade (subseção 5.2.3);
   e) as condições de trânsito consideradas para a movimentação dos assinantes na cidade (subseção 5.2.4);
   f) os parâmetros de mobilidade dos assinantes, *brokers* e publicadores (subseção 5.2.5);
   g) as métricas de desempenho adotadas para testar a hipótese do trabalho (subseção 5.2.6);
   h) a quantidade de simulações realizadas para apresentar resultados coerentes (subseção 5.2.7).

3. **Execução do experimento:** o experimento foi executado e os dados do experimento coletados (Seção 5.3).

4. **Análise dos dados:** nesta fase, a análise usada no experimento foi descrita (Seção 5.3).

5. **Interpretação dos resultados:** o passo final do experimento foi interpretar o que foi encontrado na fase de análise dos dados a luz da hipótese elaborada no início da pesquisa. Nessa interpretação, a hipótese de pesquisa foi confirmada (Seção 5.3).

5.2 Simulador

O simulador selecionado para a implementação e avaliação do modelo de comunicação baseado em eventos EMSS foi o simulador Sinalgo (DCG, 2015) (*Simulator for Network Algorithms*) devido as suas características, como: suporte à mobilidade e ao envio de eventos, documentação e tutoriais, código fonte das classes do simulador e suporte a uma linguagem de programação orientada a objetos. Sinalgo é gratuito e publicado sob

a licença BSD. Assim, o simulador foi adaptado ao cenário de uma cidade. O modelo de comunicação EMSS foi totalmente implementado, bem como o *middleware* EMSS. Aproveitou-se somente os modelos de mobilidade e a forma de conexão entre os nós, os quais já estavam implementados pelo simulador.

5.2.1 Estudos de Caso Simulados

O modelo de comunicação EMSS, implementado pelo *middleware* EMSS, foi utilizado em três estudos de caso:

1. Serviço de atendimento às situações de emergência relacionadas ao serviço de bombeiro.
2. Serviço de atendimento às situações de emergência relacionadas ao serviço policial.
3. Serviço de transporte de passageiros.

Nos três estudos de caso, os assinantes são unidades móveis e em cada estudo de caso, as simulações foram realizadas considerando três abordagens:

1. **Proativa mais apropriada (*Proactive Most Suitable - PMS*):** as unidades móveis mais adequadas foram selecionadas de acordo com os dados referentes aos valores de *fitness* de cada unidade móvel que foram atualizados proativamente, para que estivessem prontos para uso quando ocorresse um evento.
2. **Reativa mais apropriada (*Reactive Most Suitable - RMS*):** as unidades móveis mais adequadas foram selecionadas de acordo com os dados referentes ao *fitness* de cada unidade móvel que foram coletados em resposta a um chamado de evento.
3. **Reativa aleatória (*Reactive Random - RR*):** as unidades móveis foram selecionadas aleatoriamente entre as unidades móveis cinquenta por cento (50%) mais adequadas na cidade, em reação a um evento.

Essas abordagens foram usadas para testar a hipótese que conduziu o trabalho realizado de que o envio de eventos aos assinantes mais apropriados é melhor do que uma seleção aleatória entre uma porcentagem dos melhores assinantes disponíveis.

5.2.2 Área da Cidade Simulada

A área da cidade simulada foi uma área de 6 km por 6 km (36 km^2). Essa área foi dividida em 9 regiões de tamanho igual a 2 km por 2 km (4 km^2). Cada região foi

dividida em 4 células de 1 km por 1 km. No centro de cada célula, um ponto de referência foi disposto. Assim, o *broker* de cada região ficou responsável pelos chamados referentes aos 4 pontos de referência associados à sua área. Assumiu-se que em cada região existia a cobertura de um sistema de comunicação sem fio, como por exemplo, a cobertura de um sistema de telefonia celular. Considerou-se que um sistema de comunicação sem fio específico para cada região poderia atender as necessidades de comunicação em situações de emergência, frente a uma crise ou desastre natural, no caso de a infraestrutura comercial existente estar fora de operação. Entretanto, não seria um requisito obrigatório, uma vez que a cobertura do sistema de telefonia celular seria o suficiente. Por fim, o alcance de comunicação em cada região foi de 1,44 km.

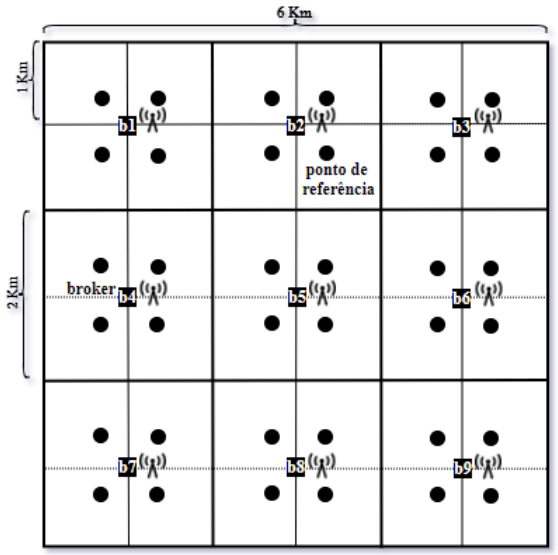


Figura 28 – Área da cidade simulada.

Fonte: Autoria Própria

Um publicador para cada um dos serviços de emergência (policial, bombeiro) foi definido para toda a cidade. Para o serviço de transporte, cada passageiro que solicitou o transporte para um destino dentro da cidade foi considerado um publicador de eventos.

5.2.3 Chamados de Eventos

Os chamados para tratar os eventos foram sinteticamente gerados seguindo um modelo de distribuição *Poisson*, o qual foi utilizado para determinar o período de tempo entre dois chamados consecutivos durante o período de 24 horas para os serviços de emergência e 12 horas para o serviço de transporte. Para o serviço de transporte foi considerado 12 horas, pois é o tempo de trabalho padrão de um motorista que pertence ao serviço. Assim, para cada serviço, foram gerados 10 históricos de chamados para os eventos. Para cada chamado gerado em cada histórico foi definido:

1. O período de tempo para o acionamento do chamado, ou seja, o período de tempo entre dois chamados consecutivos (distribuição *Poisson*).

2. O ponto de referência do evento, o qual foi escolhido aleatoriamente.

3. O *broker* ao qual o ponto de referência está associado.

4. O número de assinantes N_{si}^{e} necessários para tratar o evento, o qual também foi escolhido aleatoriamente de 1 a 3 para os serviços de emergência e 1 para o serviço de transporte.

Esses parâmetros empregados para gerar a série de chamados de eventos, nomeado de histórico de chamados, foram definidos de acordo com um *trace* real de chamadas de emergência (serviços policial e bombeiro) e de transporte. O *trace* real dos serviços de emergência foi obtido de uma cidade de médio porte do centro-oeste do Paraná durante o ano de 2016. Para o serviço de transporte, da mesma cidade, foram obtidos dados de entrevista a taxista e motorista Uber no início do ano de 2019. Os dados obtidos encontram-se de forma agregada no **Anexo I** do trabalho.

Desta forma, uma série de 12 chamadas de emergência solicitando 24 unidades móveis na média total e, com uma média de duas horas entre as chamadas foi gerada nos históricos para o serviço de bombeiro. Para o serviço policial, uma série de 9 chamadas de emergência solicitando 20 unidades móveis na média total e, com uma média de duas horas e quarenta minutos entre as chamadas nos históricos foi gerada. Para o serviço de transporte, uma série de 20 chamados de transporte para cada assinante foram gerados durante as 12 horas de sua jornada de trabalho. Os parâmetros mencionados para a criação do modelo de chamados e para a geração dos históricos nos três serviços simulados encontram-se na Tabela 6.

Além disso, para os serviços de emergência, o tempo gasto pelas unidades móveis selecionadas para atender a emergência após a chegada ao local de emergência, simplesmente chamado de **tempo de atendimento**, foi criado para cada chamada gerada, seguindo o modelo de distribuição Beta. De acordo com os dados obtidos (**Anexo I**), foi observado que o tempo máximo de atendimento a emergência $Temp_{max}$ foi de 8 horas para o serviço bombeiro e de 40 minutos para o serviço policial. O tempo mínimo de atendimento observado foi de 2 minutos e 30 segundos para o serviço de bombeiro e 20 minutos para o serviço policial. Os tempos máximo e mínimo foram utilizados para gerar os tempos de atendimento em cada serviço e encontram-se sumarizados na Tabela 6. O nível de fadiga Fad_{ij} foi cumulativo e inicializado em 0 (zero) a cada 8 horas, isso porque houve uma troca das equipes de serviço a cada 8 horas. O fator de escala do nível de fadiga, L_2, o qual é associado ao assinante foi definido com o valor 10. Utilizando o nível de fadiga no cálculo do *fitness*, outra equipe próxima a emergência menos cansada pôde ser acionada.

Tabela 6 – Parâmetros do modelo de chamados criado para a geração dos históricos.

Item	**Distribuição**	**Parâmetro**
Serviço de Bombeiro		
Total de Chamadas	Poisson	$T = 12$
Média de tempo entre as chamadas	Poisson	$I = 2h$
Média de unidades móveis solicitadas	Randômica	$M = 24$
Localização do Evento	Randômica	$R = \{r_1, \ldots, r_{36}\}$
Tempo de Atendimento	Beta	$x_{min} = 2'$ e 30"; $x_{max} = 8h$
Serviço Policial		
Total de Chamadas	Poisson	$T = 9$
Média de tempo entre as chamadas	Poisson	$I = 2h$ e $40'$
Média de unidades móveis solicitadas	Randômica	$M = 12$
Localização do Evento	Randômica	$R = \{r_1, \ldots, r_{36}\}$
Tempo de Atendimento	Beta	$x_{min} = 20'$; $x_{max} = 40'$
Serviço de Transporte		
Chamado por Unidade Móvel	Poisson	$T = 20$
Média de tempo entre as chamadas	Poisson	$I = 37'$
Localização do Usuário	Randômica	$R = \{r_1, \ldots, r_{36}\}$
Localização de Destino	Randômica	$R = \{r_1, \ldots, r_{36}\}$
Tempo de Desembarque	Beta	$x_{min} = 2'$; $x_{max} = 5'$

Fonte: Autoria Própria

Para o serviço de transporte, o **tempo de desembarque** após o assinante chegar ao local de destino, seguiu o modelo de distribuição Beta. O **tempo de desembarque** é o tempo que o passageiro demora para sair da unidade móvel quando chega ao seu destino. Esse tempo inclui pagar a viagem e pegar as bagagens, quando estas existem. Esse tempo foi gerado com base em dados empíricos. Assim, assumiu-se um tempo de desembarque distribuído entre 2 e 5 minutos (Tabela 6). O nível de fadiga Fad_{ij} foi inicializado em 0 (zero) e foi cumulativo. O nível de avaliação do motorista $Class_j$ foi definido como $\{0, 1, 2, 3, 4, 5\}$ e L_3 que é o fator de escala relacionado ao nível de classificação associado ao assinante foi definido em 10. Os assinantes receberam um nível de avaliação fixo durante toda a simulação para mostrar o comportamento das unidades móveis em relação ao nível de avaliação do motorista.

O período de tempo entre um chamado de evento e outro, o **tempo de atendimento** e o **tempo de desembarque** foram gerados no Matlab em arquivos de texto, os quais foram lidos posteriormente pelo simulador. Para cada serviço, arquivos com os tempos de atendimento e desembarque para os chamados gerados durante 24 horas ou nas 12 horas do dia, no caso do serviço de transporte, foram gerados. Para fins de comparação entre as abordagens, para cada número de simulação em cada abordagem e serviço, os mesmos tempos de atendimento e desembarque foram utilizados para cada chamado. Em

Tabela 7 – Definição do coeficiente de tráfego c_{ij}^{k}.

Tráfego	**Coeficiente de Tráfego**
Lento	Entre 0 e 0,30 (inclusive)
Moderado	Entre 0,30 e 0,60 (inclusive)
Intenso	Entre 0,60 e 1,0 (exclusive)

Fonte: Autoria Própria

cada abordagem de cada serviço, as simulações iniciaram pelo número 1. Por exemplo, no serviço policial, para a simulação 1 da abordagem PMS com 3 unidades móveis, RMS com 3 unidades móveis e RR com 3 unidades móveis, a primeira chamada do dia para o serviço policial teve um tempo de atendimento de 21 minutos que foi utilizada em todas as abordagens, a segunda 20 minutos e 15 segundos para todos as abordagens e, assim, sucessivamente. O mesmo arquivo de atendimento foi utilizado 3 vezes e cada arquivo conteve um tempo de atendimento para cada chamado do dia.

5.2.4 Condições de Trânsito

O simulador assumiu que a velocidade das unidades móveis variou entre 30 km/h e 60 km/h e, em relação às condições de tráfego, considerou-se tráfego como lento, moderado ou intenso. Desta forma, o coeficiente de tráfego c_{ij}^{k} sofreu algumas variações conforme mostra a Tabela 7. Cabe salientar, que dependendo do horário do dia o trânsito muda e, como consequência, o coeficiente de trânsito também. Embora o trânsito seja intenso, moderado ou leve, o coeficiente sofre variações dentro dessas condições. Os valores do coeficiente de tráfego foram usados para ajustar o valor do *fitness* de modo que com trânsito leve, o valor do *fitness* fosse maior do que com trânsito moderado ou intenso, considerando um mesmo valor de distância. No caso de trânsito moderado, o valor do *fitness* deve ficar maior do que com trânsito intenso e menor do que com trânsito leve. Assim, o *fitness* é influenciado pelas condições de trânsito da cidade. As condições de trânsito utilizadas levaram em consideração os dados obtidos de http://www.cetsp.com.br/media/574933/2016volumevelocidade.pdf. O horário de pico foi considerado como trânsito intenso. A Tabela 8 mostra as condições de trânsito de uma cidade real, as quais foram adotadas no trabalho.

5.2.5 Parâmetros do Simulador

Alguns parâmetros do simulador foram configurados. O modelo de mobilidade dos *brokers* e do(s) publicador(es) adotado foi sem mobilidade (*NoMobility*). As unidades móveis utilizaram uma extensão da classe *PerfectRWP*, na qual as unidades móveis selecionam um ponto aleatório da área de simulação (algumas vezes, o ponto da situação de emergência

Tabela 8 – Condições de trânsito de uma cidade real.

Período do Dia	**Condições de Trânsito**
1 a.m. - 7 a.m.	Leve
7 a.m. - 10 a.m.	Intenso
10 a.m. - 12 a.m.	Moderado
12 a.m. - 14 p.m.	Intenso
14 p.m. - 15 p.m.	Moderado
15 p.m. - 16 p.m.	Leve
16 p.m. - 17 p.m.	Moderado
17 p.m. - 20 a.m.	Intenso
20 p.m. - 1 a.m.	Leve

Fonte: Autoria Própria

ou do chamado do passageiro) e se movimentam até ele. Os nós, isto é, os *brokers*, as unidades móveis e o publicador puderam enviar mensagens enquanto as estavam recebendo. Os *brokers* foram distribuídos em grade e as unidades móveis em pontos fixos na área de simulação, sendo sempre dispostas exatamente nos mesmos locais no momento inicial das simulações. Para os serviços de emergência, as unidades móveis foram distribuídas entre os *brokers*, os quais estavam distribuídos na cidade, a saber:

- **Conjunto 1:** 3 unidades móveis no total, as quais foram dispostas nos *brokers* b_1, b_5 e b_9, respectivamente.
- **Conjunto 2:** 6 unidades móveis no total, as quais foram dispostas nos *brokers* b_1, b_5, b_9, b_3, b_7 e b_8, respectivamente.
- **Conjunto 3:** 9 unidades móveis no total, as quais foram dispostas nos *brokers* b_1, b_5, b_9, b_3, b_7, b_8, b_2, b_4 e b_6, respectivamente.
- **Conjunto 4:** 12 unidades móveis no total, as quais foram dispostas nos *brokers* b_1, b_5, b_9, b_3, b_7, b_8, b_2, b_4, b_6, b_1, b_5 e b_9, respectivamente.
- **Conjunto 5:** 15 unidades móveis no total, as quais foram dispostas nos *brokers* b_1, b_5, b_9, b_3, b_7, b_8, b_2, b_4, b_6, b_1, b_5, b_9, b_3, b_7 e b_8, respectivamente.

Para o serviço de transporte, as unidades móveis foram distribuídas entre os *brokers* que estavam distribuídos na cidade:

- **Conjunto 1:** 6 unidades móveis no total, as quais foram dispostas nos brokers b_1, b_5, b_9, b_3, b_7 e b_8, respectivamente.

- **Conjunto 2:** 12 unidades móveis no total, as quais foram dispostas nos brokers b_1, b_5, b_9, b_3, b_7, b_8, b_2, b_4, b_6, b_1, b_5 e b_9, respectivamente.
- **Conjunto 3:** 18 unidades móveis no total, duas por *broker*.
- **Conjunto 4:** 24 unidades móveis no total, duas por *broker*, e ainda mais 6 unidades móveis dispostas nos *brokers* b_1, b_5, b_9, b_3, b_7 e b_8, respectivamente.

Outro parâmetro de simulação importante foi o intervalo das mensagens *bind* que foi fixado em 1 minuto durante toda a simulação. Entre o intervalo de cada mensagem *bind*, as condições de tráfego podem alterar significativamente o valor de *fitness*. Para a definição do valor de intervalo de *bind*, testes foram realizados. Com isso, foi concluído que esse tempo é o apropriado porque com um valor menor, o tráfego na rede aumenta e com um valor maior a distância que a unidade móvel pode se deslocar é significativa e pode aumentar o tempo para que a unidade móvel chegue até o evento.

5.2.6 Métricas de Desempenho

A métrica adotada para comparar o desempenho das abordagens PMS, RMS e RR em todos os serviços para testar a hipótese do trabalho foi:

- **Tempo de Resposta do Serviço (*Service Response Time*) - TRS:** o período de tempo desde o momento em que um chamado para o evento acontece até o momento em que cada unidade móvel solicitada chega ao local do evento. Para os serviços de emergência (bombeiro e policial), o local do evento é o local da emergência. Para o serviço de transporte, o local do evento é o local em que o passageiro se encontra em que a unidade móvel deve apanhá-lo para levá-lo ao seu destino.

Para o serviço de transporte, além do TRS em função do **número de unidades disponíveis** e em função do **nível de avaliação do motorista**, a seguinte métrica foi utilizada:

- **Número médio de unidades móveis ativadas (*Average Number of Mobile Units Activated*) - NUMA:** O número médio de unidades móveis ativadas em 12 horas de acordo com o **nível de avaliação do motorista**. Este número significa a média de unidades móveis ativadas pertencentes a um dado **nível de avaliação do motorista** nos históricos de chamados simulados.

O TRS inclui o tempo necessário para selecionar a(s) unidade(s) móvel(is) (de acordo com cada uma das abordagens) mais o tempo para cada unidade móvel se deslocar da localização na cidade quando foi ativada até o local do evento. Para o serviço policial

e de transporte, as unidades móveis poderiam ser ativadas enquanto se deslocavam pela cidade, enquanto que no serviço de bombeiro, as unidades móveis só deveriam ser ativadas a partir de suas bases.

5.2.7 Número de Simulações

Para os serviços de emergência, as três abordagens foram comparadas considerando o TRS de acordo com o cenário da Tabela 9. Para cada histórico de chamado gerado, 30 rodadas de simulação foram executadas. Assim, com os 5 cenários da Tabela 9 simulados para cada uma das abordagens PMS, RMS e RR, 4.500 rodadas de simulação foram realizadas para cada um dos serviços de emergência, ou seja, 9.000 rodadas de simulação no total.

Tabela 9 – Cenários simulados nos serviços de emergência.

Cenário	**Unidade Móveis**	**Chamadas/Dia***	**RS**	Número de HC	TRS
1	3	12-9	30	10	300
2	6	12-9	30	10	300
3	9	12-9	30	10	300
4	12	12-9	30	10	300
5	15	12-9	30	10	300

*Média de chamadas por dia para bombeiro-polícia RS = Rodadas de Simulação HC = Histórico de chamados TRS = Total de Rodadas de Simulação

Fonte: Autoria Própria

Para o serviço de transporte, as abordagens foram comparadas considerando o TRS e o NUMA de acordo com o cenário da Tabela 10. Com os 4 cenários simulados (Tabela 10) para cada uma das abordagens PMS, RMS e RR, 3.600 rodadas de simulação foram realizadas para este serviço.

Tabela 10 – Cenários simulados no serviço de transporte.

Cenário	**Unidade móveis**	**Chamados/dia***	**RS**	Número de HC	TRS
1	6	120	30	10	300
2	12	240	30	10	300
3	18	360	30	10	300
4	24	480	30	10	300

*Total de chamados por dia RS = Rodadas de Simulação HC = Histórico de chamados TRS = Total de Rodadas de Simulação

Fonte: Autoria Própria

Cabe salientar que os *brokers* se mantiveram sempre conectados e em comunicação durante as simulações. Desta forma, a questão de tolerância a faltas não foi tratada no trabalho.

5.3 Resultados

Para os resultados várias simulações foram realizadas. Os resultados das simulações para o serviço de bombeiro, policial e de transporte encontram-se nesta seção. O nível de confiança calculado foi de 95% em todos os serviços.

5.3.1 Serviço de Bombeiro

Para o serviço bombeiro, as unidades móveis somente se movimentam quando precisam atender a uma chamada de emergência, sendo acionadas de seus postos de atendimento ou bases. Por esta razão apresentam mobilidade parcial.

5.3.1.1 Comparação dos Tempos de Resposta do Serviço

Cada unidade móvel é acionada com base em seu valor de *fitness*. Quando acionada, a unidade móvel se desloca até a emergência percorrendo um determinado trajeto em um certo tempo. Desta forma, os resultados obtidos do **Tempo de Resposta do Serviço** em relação ao número de unidades móveis disponíveis nos cenários apresentados na Seção 5.2.7, Tabela 9, para as abordagens PMS, RMS e RR são mostrados na Figura 29. Na figura, o TRS é apresentado em minutos. Em cada abordagem, pode ser observado que o TRS diminuiu significativamente quando o número de unidades móveis disponíveis aumentou. Assim, comparando as abordagens entre si, as abordagens PMS e RMS apresentaram um TRS menor do que a abordagem aleatória (RR). O TRS é maior quando é feita uma escolha aleatória, independentemente do número de unidades móveis disponíveis. Nas abordagens PMS e RMS, o TRS ficou muito próximo, pois as unidades móveis são acionadas a partir de seus postos de serviço. No entanto, a abordagem RMS mostrou um TRS sutilmente inferior à abordagem PMS, conforme o número de unidades móveis foi aumentando.

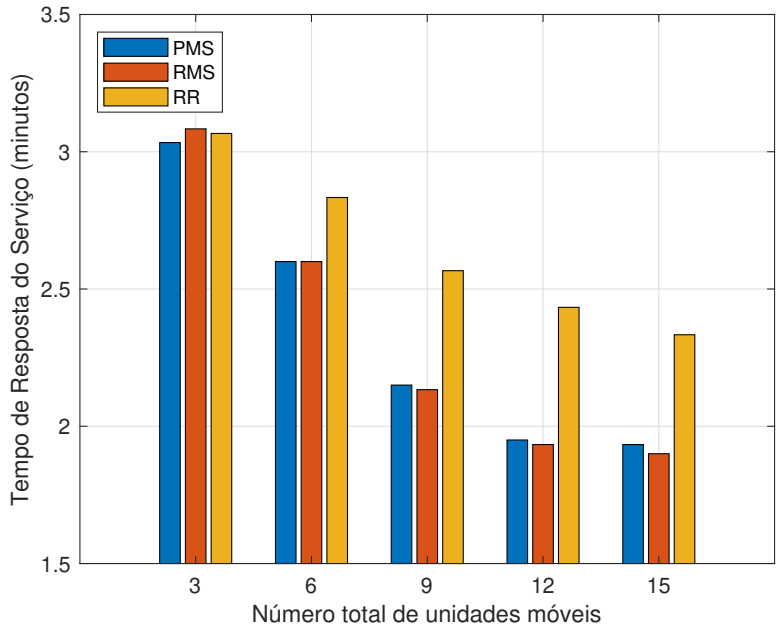


Figura 29 – Comparação do TRS nas abordagens PMS, RMS e RR para o serviço de bombeiro.

Fonte: Autoria Própria

A Figura 30 mostra a média dos valores de *fitness* calculados para acionar as unidades móveis mais apropriadas para tratar cada emergência. Da mesmna forma, a Tabela 11 apresenta a média dos valores máximo e mínimo do *fitness* quando os *brokers* foram acionados e realizaram a escolha das unidades móveis mais apropriadas. Pode-se observar na figura que a média dos valores de *fitness* são praticamente iguais nas abordagens PMS e RMS. O *fitness* leva em consideração vários fatores, dentre eles a distância. Este último pode ter feito com que as duas abordagens apresentassem resultados muito próximos em relação ao TRS. A velocidade das unidades móveis, a qual considera as condições de trânsito, enquanto estavam se deslocando até cada emergência pode ter influenciado o TRS.

A Tabela 12, mostra a média das distâncias percorridas pelas unidades móveis que foram selecionadas para o atendimento às emergências, de acordo com as médias dos valores de *fitness* da Figura 30. Os valores de *fitness* da abordagem PMS e RMS são muito próximos, bem como as distâncias percorridas, o que resultou em um TRS muito próximo entre as duas abordagens.

Para o serviço de bombeiro, em todas as abordagens analisadas escolher as unidades móveis mais adequadas foi melhor do que uma escolha aleatória dentro de um subconjunto das unidades móveis mais apropriadas. Além disso, a abordagem RMS pode ser recomendada para este serviço que apresenta mobilidade parcial, uma vez que apresentou um **Tempo de Resposta do Serviço** sutilmente menor em relação a abordagem PMS,

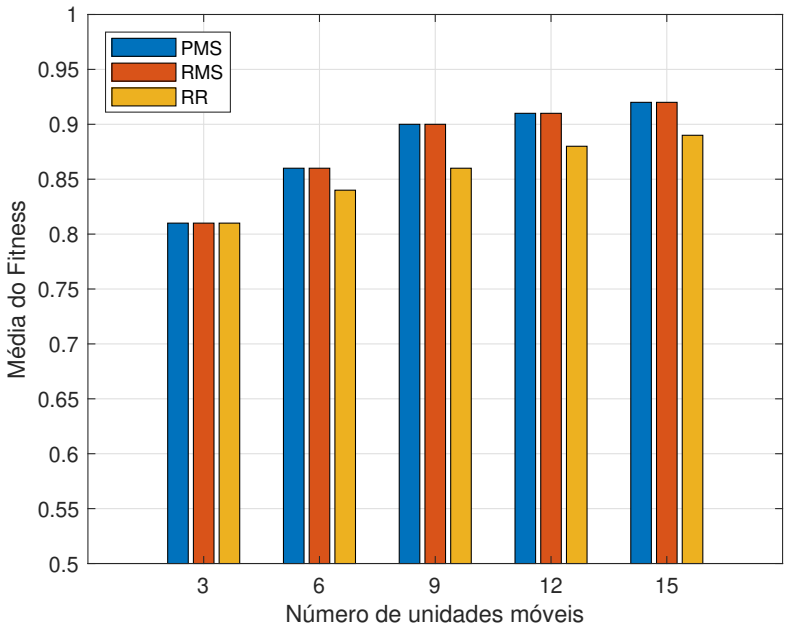


Figura 30 – Média dos valores de *fitness* para o serviço de bombeiro.

Fonte: Autoria Própria

Tabela 11 – Médias dos valores de *fitness* máximo e mínimo para o serviço de bombeiro.

	PMS	**RMS**	**RR**
Unidades móveis	**F-min - F-max**	**F-min - F-max**	**F-min - F-max**
3	0,71 - 0,88	0,68 - 0,86	0,74 - 0,86
6	0,63 - 0,92	0,61 - 0,91	0,70 - 0,91
9	0,61 - 0,94	0,60 - 0,93	0,69 - 0,93
12	0,63 - 0,94	0,61 - 0,94	0,71 - 0,94
15	0,65 - 0,94	0,63 - 0,94	0,74 - 0,94

F-min = *Fitness* mínimo F-max = *Fitness* máximo

Fonte: Autoria Própria

e muito inferior a abordagem RR.

5.3.2 Serviço Policial

No serviço policial, as unidades móveis se movimentam pela cidade e param eventualmente em locais que precisam de atenção. As unidades móveis podem ser ativadas enquanto se movimentam pela cidade. Logo, este serviço apresenta mobilidade.

Tabela 12 – Médias das distâncias percorridas para tratar chamadas de emergência no serviço de bombeiro.

	PMS	**RMS**	**RR**
Unidades móveis	**Distância**	**Distância**	**Distância**
3	2,42	2,37	2,45
6	2,05	2,04	2,27
9	1,73	1,71	2,07
12	1,59	1,57	1,97
15	1,59	1,55	1,90

A Distância é fornecida em Km.

Fonte: Autoria Própria

5.3.2.1 Comparação dos Tempos de Resposta do Serviço

A Figura 31 mostra o comportamento do TRS em termos do número de unidades móveis disponíveis para os cenários apresentados na Seção 5.2.7, Tabela 9, para as abordagens PMS, RMS e RR. O **Tempo de Resposta do Serviço** é apresentado na figura em minutos. Nos dois casos em que as unidades móveis mais adequadas foram escolhidas (PMS e RMS), o TRS diminuiu significativamente quando o número de unidades móveis disponíveis aumentou. O TRS é maior quando uma escolha aleatória (RR) é realizada, independentemente do número de unidades móveis disponíveis.

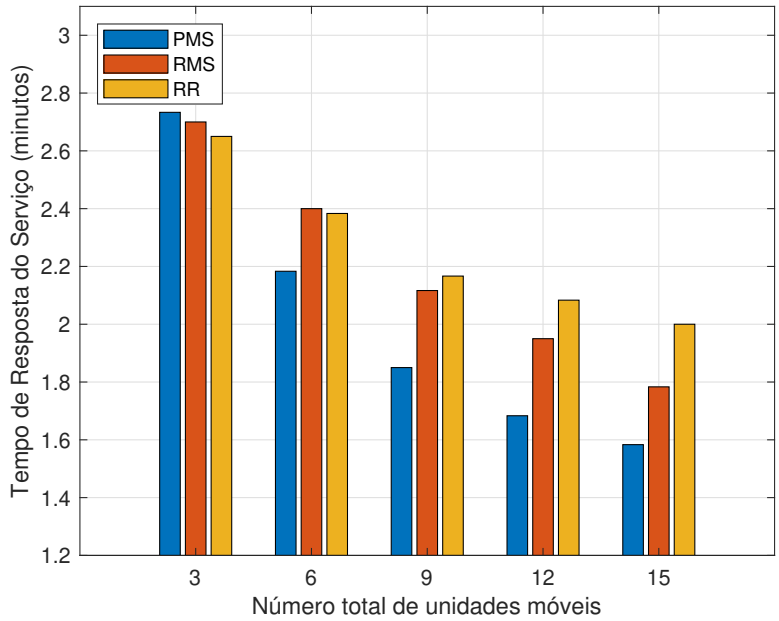


Figura 31 – Comparação do TRS nas abordagens PMS, RMS e RR para o serviço policial.

Fonte: Autoria Própria

A Figura 31 possui relação com a Tabela 13, a qual mostra a média das distâncias do serviço policial, e com a Figura 32 que mostra as médias dos valores de *fitness* das unidades móveis acionadas para o serviço. A Tabela 14 ilustra a média dos valores máximo e mínimo do *fitness* quando os *brokers* foram acionados e realizaram a escolha das unidades móveis mais apropriadas.

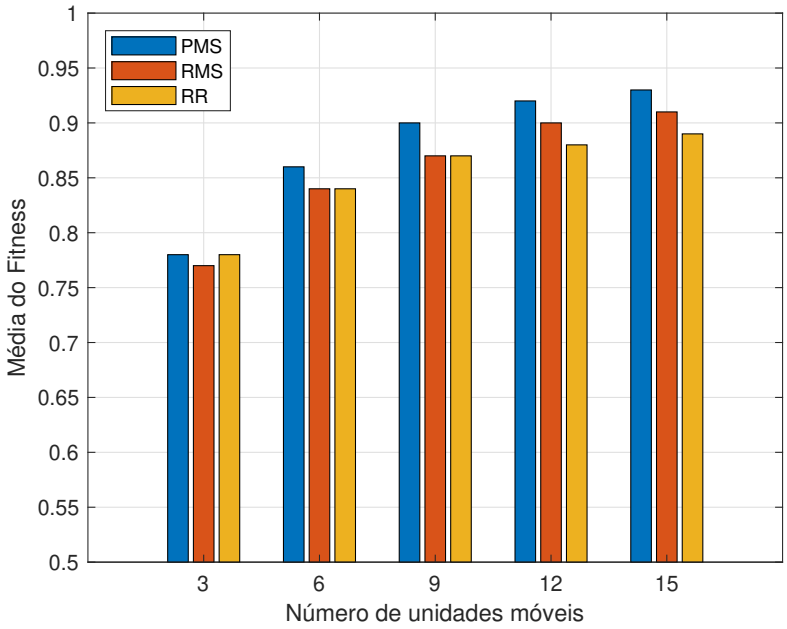


Figura 32 – Média dos valores de *fitness* das abordagens PMS, RMS e RR para o serviço policial.

Fonte: Autoria Própria

Levando-se em consideração os dados da Tabela 13, pode-se concluir que quanto mais unidades móveis disponíveis para atendimento, menor é a distância a ser percorrida e quanto maior é o valor do *fitness* (Figura 32), mais a distância diminui. Isso ocorre em todas as abordagens. Cabe salientar que a distância e o tempo de atendimento, o qual é utilizado para o cálculo do nível de fadiga, possuem relação direta com o valor do *fitness* calculado e, por consequência, na escolha das unidades móveis. Quanto mais próxima a unidade móvel do ponto de referência, maior é o seu *fitness* e quanto maior o tempo de atendimento maior será o nível de fadiga e menor o *fitness*. A média do nível de fadiga não é apresentada pois o nível de fadiga é zerado a cada 8 horas em cada simulação.

De acordo com os dados e os gráficos apresentados, escolher as unidades móveis mais adequadas foi melhor do que uma escolha aleatória em todas as abordagens analisadas. Ainda, a abordagem proativa para o serviço policial se mostrou melhor do que as outras duas abordagens. Assim, a abordagem PMS pode ser recomendada para o serviço policial que apresenta mobilidade.

Tabela 13 – Médias das distâncias percorridas para tratar chamadas de emergência no serviço policial.

	PMS	**RMS**	**RR**
Unidades móveis	**Distância**	**Distância**	**Distância**
3	2,17	2,12	2,11
6	1,76	1,92	1,90
9	1,51	1,72	1,75
12	1,38	1,58	1,69
15	1,32	1,47	1,63

A Distância é dada em Km.

Fonte: Autoria Própria

Tabela 14 – Médias dos *Fitness* máximo e mínimo para o serviço policial.

	PMS	**RMS**	**RR**
Unidades móveis	**F-min - F-max**	**F-min - F-max**	**F-min - F-max**
3	0,71 - 0,82	0,72 - 0,79	0,76 - 0,80
6	0,70 - 0,90	0,72 - 0,87	0,78 - 0,87
9	0,69 - 0,93	0,72 - 0,90	0,79 - 0,91
12	0,69 - 0,95	0,71 - 0,92	0,80 - 0,92
15	0,70 - 0,95	0,71 - 0,93	0,80 - 0,93

F-min = *Fitness* mínimo F-max = *Fitness* máximo

Fonte: Autoria Própria

5.3.3 Serviço de Transporte

No serviço de transporte a mobilidade está presente, pois as unidades móveis se movimentam pela cidade e param para pegar passageiros e os deixar em seu destino. Da mesma forma que no serviço policial, as unidades móveis podem ser ativadas enquanto se movimentam pela cidade.

5.3.3.1 Comparação do Número de Unidades Móveis Ativadas

Os 4 cenários apresentados na Seção 5.2.7, Tabela 10, foram simulados para as abordagens PMS, RMS e RR. Todos mostraram um comportamento semelhante quando avaliou-se o **número de unidades móveis ativadas** em relação ao **nível de avaliação do motorista**. A Figura 33 mostra um exemplo deste comportamento para o cenário 4. Na figura, pode-se verificar que o *fitness* leva em conta o **nível de avaliação do motorista**, pois motoristas com uma avaliação maior são acionados com mais frequência. Isso aconteceu em todas as abordagens.

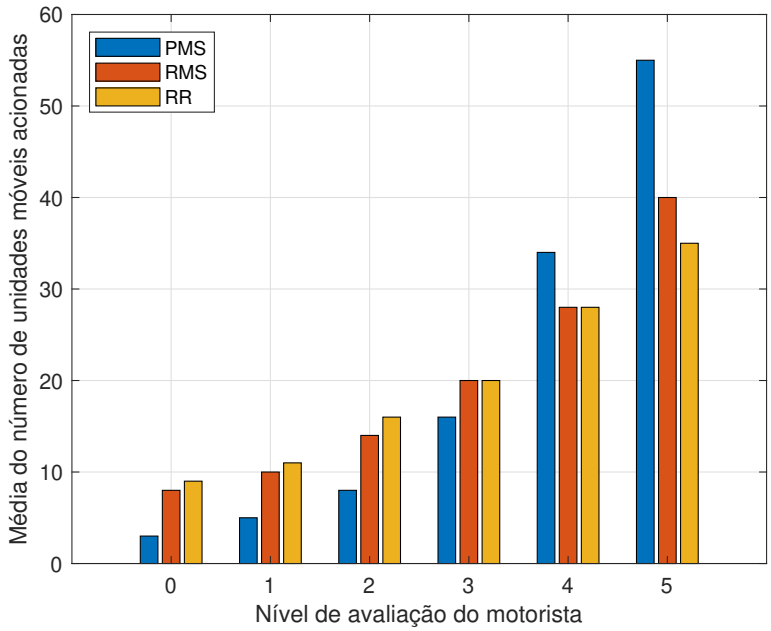


Figura 33 – Média do número de unidades móveis ativadas de acordo com o nível de avaliação do motorista para o serviço de transporte.

Fonte: Autoria Própria

5.3.3.2 Comparação dos Tempos de Resposta do Serviço

A comparação do TRS para o serviço de transporte foi analisada em função do **número de unidades móveis disponíveis** e do **nível de avaliação do motorista** considerando os cenários da Seção 5.2.7, Tabela 10.

a) **Comparação do TRS em relação ao Número de Unidades Móveis Disponíveis**

A Figura 34 mostra o comportamento do TRS levando em consideração o número de unidades disponíveis. Na figura, o TRS foi menor na abordagem PMS em relação as outras abordagens em todos os cenários simulados. Na abordagem PMS, o TRS diminuiu a medida em que o número de unidades móveis aumentou. Isso quer dizer que com mais unidades móveis disponíveis, mais rapidamente as mesmas chegam ao local em que o passageiro se encontra. Na abordagem RMS, o TRS apresentou um comportamento similar. A abordagem RR não apresentou um padrão definido pois escolhe as unidades móveis de forma aleatória. A diferença do TRS, especialmente entre as abordagens RMS e PMS, se deve ao tempo gasto para que a abordagem reativa tenha condições de fazer a escolha de quais unidades móveis devem ser acionadas.

Para chegar ao local onde o passageiro se encontra, com as médias de tempo apresentadas na Figura 34, uma certa distância foi percorrida pelas unidades móveis.

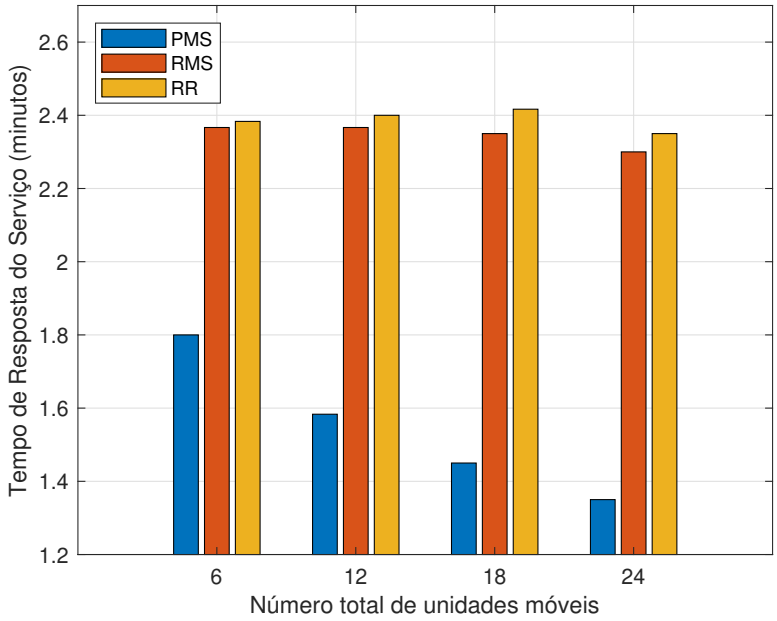


Figura 34 – Comparação das abordagens PMS, RMS e RR em relação ao número de unidades móveis disponíveis para o serviço de transporte.

Fonte: Autoria Própria

Assim, a Tabela 15 apresenta a média das distâncias percorridas pelas unidades móveis acionadas na figura.

Tabela 15 – Médias das distâncias percorridas para tratar chamadas de passageiros no serviço de transporte.

	PMS	**RMS**	**RR**
Unidades móveis	**Distância**	**Distância**	**Distância**
6	1,53	1,98	2,00
12	1,38	1,96	1,99
18	1,27	1,94	1,99
24	1,20	1,92	1,95

A Distância é dada em Km.

Fonte: Autoria Própria

Para a escolha das unidades móveis, as quais percorreram a média das distâncias que estão na Tabela 15, valores de *fitness* foram utilizados. Desta forma, a Figura 35 mostra a média dos valores de *fitness* utilizados para a seleção daquelas unidades móveis e a Tabela 16 apresenta a média dos valores mínimos e máximos de *fitness* quando a escolha das unidades móveis foi realizada nos cenários apresentados. A abordagem PMS apresentou uma média de valores de *fitness* maior em relação as outras abordagens. Um

fitness maior representa uma distância menor a ser percorrida e, por vezes, em um tempo menor.

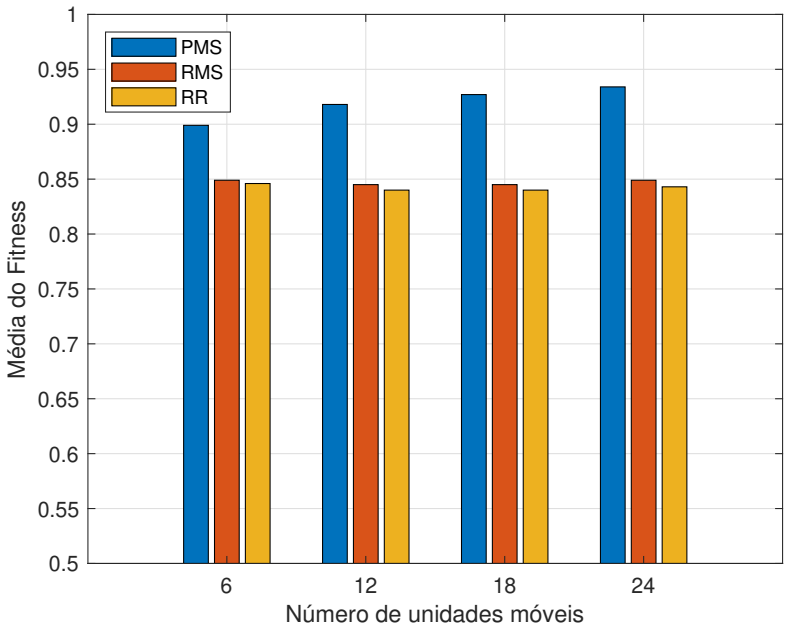


Figura 35 – Comparação do *fitness* para as abordagens PMS, RMS e RR em relação ao número de unidades móveis disponíveis para o serviço de transporte.

Fonte: Autoria Própria

Tabela 16 – Média dos valores mínimo e máximo de *fitness* para o serviço de transporte.

	PMS	**RMS**	**RR**
Unidades móveis	**F-min - F-max**	**F-min - F-max**	**F-min - F-max**
6	0,696 - 0,916	0,811 - 0,840	0,826 - 0,856
12	0,701 - 0,926	0,817 - 0,855	0,821 - 0,854
18	0,721 - 0,938	0,829 - 0,857	0,824 - 0,845
24	0,705 - 0,934	0,820 - 0,849	0,819 - 0,849

F-min = *Fitness* mínimo F-max = *Fitness* máximo

Fonte: Autoria Própria

Por fim, de acordo com os dados apresentados, escolher as unidades móveis mais adequadas é melhor do que uma escolha aleatória. Os motoristas chegam mais rapidamente até o passageiro e, como consequência, até o destino. Além disso, a abordagem proativa apresentou resultados melhores em relação as outras duas abordagens simuladas, podendo ser utilizada para este serviço que apresenta mobilidade.

b) Comparação do TRS em relação ao Nível de Avaliação do Motorista

Dos 4 cenários simulados, todos mostraram um comportamento semelhante. Logo, somente o cenário 4 será analisado. A Figura 36 mostra o TRS para as abordagens PMS, RMS e RR em função do **nível de avaliação do motorista**. A abordagem PMS mostrou um desempenho melhor do que as abordagens RMS e RR, pois apresentou um TRS menor. Na figura o TRS é apresentado em minutos.

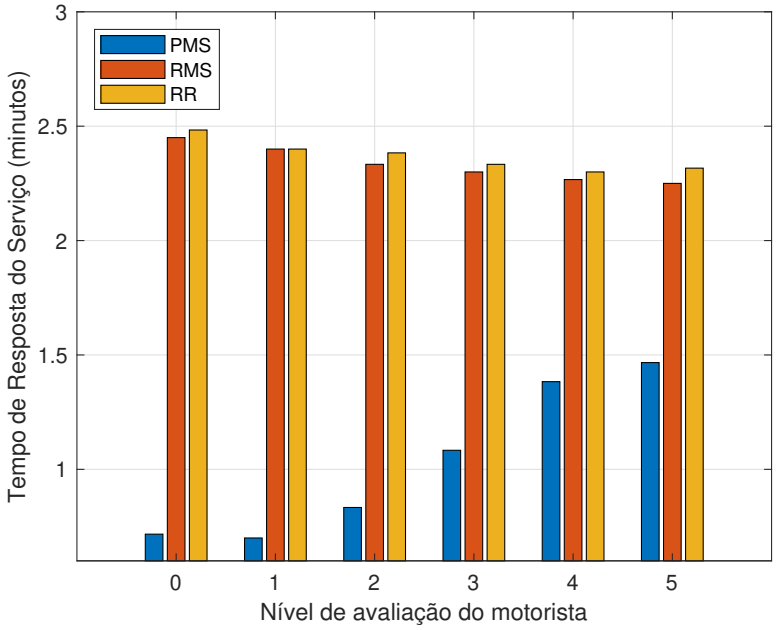


Figura 36 – **Tempo de Resposta do Serviço** das abordagens PMS, RMS e RR para o serviço de transporte.

Fonte: Autoria Própria

A Tabela 17 mostra, de acordo com o **nível de avaliação do motorista**, a média das distâncias percorridas pelas unidades móveis acionadas na Figura 33. Essa figura tem relação com o cenário que está sendo apresentado. A tabela mostra que a média das distâncias aumentou gradualmente e suavemente conforme o **nível de avaliação do motorista** aumentou, pois motoristas com um número de avaliação maior são acionados mais vezes.

Para o acionamento das unidades móveis pelos *brokers*, os valores de *fitness* das mesmas são necessários. Assim, a Figura 37 mostra a média dos valores de *fitness* das unidades móveis acionadas nas abordagens PMS, RMS e RR (Figura 33).

De acordo com os dados apresentados, realizar a escolha das unidades móveis mais adequadas é melhor do que uma escolha aleatória. Além disso, a abordagem PMS pode ser indicada para esse serviço porque escolheu o melhor motorista baseado no **nível de avaliação do motorista** e apresentou um TRS menor em relação às outras abordagens. Com isso, o passageiro viaja em um tempo menor e, como consequência, percorre uma distância menor. A métrica de *fitness* permite isso.

Tabela 17 – Média das distâncias percorridas nas abordagens RMS e RR para o serviço de transporte.

Cenário 4	**Média das distâncias (Km)**					
Nível de avaliação do motorista	**0**	**1**	**2**	**3**	**4**	**5**
PMS	0.47	0.46	0.59	0.89	1.24	1.35
RMS	1.85	1.85	1.86	1.90	1.93	1.96
RR	1.88	1.88	1.92	1.94	1.96	2.01

Fonte: Autoria Própria

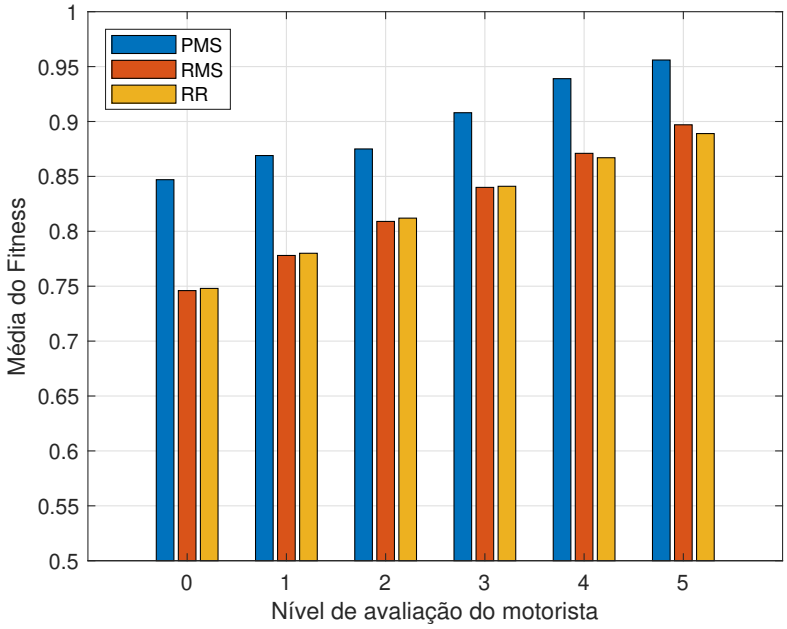


Figura 37 – Média do *fitness* nas abordagens PMS, RMS e RR para o serviço de transporte.

Fonte: Autoria Própria

5.3.4 Comparação do Número de Mensagens

Em relação ao número de mensagens trocadas entre as entidades do *middleware*, uma avaliação foi realizada em todos os serviços. A Tabela 18 mostra de forma resumida o comportamento do número total de mensagens trocadas entre as entidades do *middleware* para uma única simulação considerando as abordagens PMS, RMS e RR.

Tabela 18 – Visão do número total de mensagens trocadas nas diferentes abordagens para uma única simulação.

Abordagem/Mensagem	**PMS**	**RMS**	**RR**
notification-system	Número de acionamentos na simulação somado pela quantidade de vezes que pedido não foi atendido	Número de acionamentos na simulação somado pela quantidade de vezes que pedido não foi atendido	Número de acionamentos na simulação somado pela quantidade de vezes que pedido não foi atendido
ack notification-system	Número de *notification-system* emitidos	Número de *notification-system* emitidos	Número de *notification-system* emitidos
notification	Número de notificações realizadas na simulação	Número de notificações realizadas na simulação	Número de notificações realizadas na simulação
invite-subscribe	-	Número de acionamentos na simulação somado pela quantidade de vezes que pedido não foi atendido multiplicados pelo número de *brokers* intermediários	Número de acionamentos na simulação somado pela quantidade de vezes que pedido não foi atendido multiplicados pelo número de *brokers* intermediários
subscribe	Número de vezes que unidade móvel muda de região	Número de vezes que *broker* publicador é acionado somado ao número de vezes que *broker* intermediário é acionado	Número de vezes que *broker* publicador é acionado somado ao número de vezes que *broker* intermediário é acionado
bind	Número de minutos que a unidade móvel permancece em uma região menos o tempo de atendimento e o número de mensagens *subscribe*	Número de notificações realizadas na simulação	Número de notificações realizadas na simulação
unsubscribe	Número de vezes que a unidade móvel muda de região	-	-
ack-notification	Número de notificações realizadas na simulação	Número de notificações realizadas na simulação	Número de notificações realizadas na simulação

Abordagem/Mensagem	**PMS**	**RMS**	**RR**
publish	Número de vezes que *broker* publicador não tem unidades móveis disponíveis e precisa solicitar para algum *broker* intermediário	Número de vezes que *broker* publicador não tem unidades móveis disponíveis e precisa solicitar para algum *broker* intermediário	Número de vezes que *broker* publicador não tem unidades móveis disponíveis e precisa solicitar para algum *broker* intermediário
ack-publish	Número de mensagens *publish* emitidas	Número de mensagens *publish* emitidas	Número de mensagens *publish* emitidas
invite-updated-fitness	-	Número de acionamentos na simulação somado pela quantidade de vezes que pedido não foi atendido	Número de acionamentos na simulação somado pela quantidade de vezes que pedido não foi atendido
updated-fitness	Número de mensagens *subscribe* somadas as mensagens de *bind*, *unsubscribe* e *ack-notification* recebidas em cada *broker*	Número de mensagens *subscribe* somadas as mensagens de *bind*, *unsubscribe* e *ack-notification* recebidas em cada *broker*	Número de mensagens *subscribe* somadas as mensagens de *bind*, *unsubscribe* e *ack-notification* recebidas em cada *broker*

Fonte: Autoria Própria

Quanto aos serviços com mobilidade (i. e., policial e de transporte de passageiros), os seguintes dados foram observados:

- O número de mensagens *subscribe* foi em média 7 vezes maior na abordagem proativa em relação a abordagem reativa.
- O número de mensagens *unsubscribe* foi em média 8 vezes maior na abordagem proativa em relação a abordagem reativa.
- O número de mensagens para atualização de estado (*update-fitness*) foi em média 744 vezes maior na abordagem proativa em relação a abordagem reativa.
- O número de mensagens *notification-system* foi em média 5 vezes maior na abordagem reativa em relação a abordagem proativa e, como consequência o número de mensagens *ack-notification-system* também.
- As mensagens *publish*, *ack-publish*, *notification* e *ack-notification* se mantiveram iguais tanto na abordagem reativa quanto na abordagem proativa.

Desta forma, considerando os serviços com mobilidade, em média o número total de mensagens trocadas entre as entidades do *middleware* foi 224 vezes maior na abordagem proativa que nas abordagens reativas (RMS e RR). O gráfico da Figura 38 mostra o número total de mensagens trocadas entre as entidades do *middleware* para o serviço policial na abordagem PMS. O mesmo comportamento foi observado no serviço de transporte.

Na Figura 39, o número de mensagens trocadas entre as entidades do *middleware* nas abordagens RMS e RR para o serviço policial é apresentado. O número elevado de mensagens quando o número de unidades móveis foi igual a 3 aconteceu especialmente devido a questão de que, muitas vezes, quando um evento ocorreu não havia unidades móveis disponíveis. Como consequência, as tentativas para que a ocorrência fosse atendida fizeram com que o número de mensagens aumentasse em relação aos outros cenários. Nos outros cenários que apresentaram mais unidades móveis disponíveis, as tentativas de atendimento foram basicamente nulas. Nas figuras 38 e 39 pode-se perceber um crescimento linear do número de mensagens conforme o número de unidades móveis aumentou.

Em relação ao serviço com mobilidade parcial (i. e., bombeiro), o número total de mensagens foi em média 238 vezes maior na abordagem proativa em relação a abordagem reativa. O gráfico da Figura 40 mostra o número total de mensagens da abordagem proativa. A Figura 41 ilustra a média do número total de mensagens entre as abordagens RMS e RR.

Dos totais apresentados para cada cenário, observou-se que:

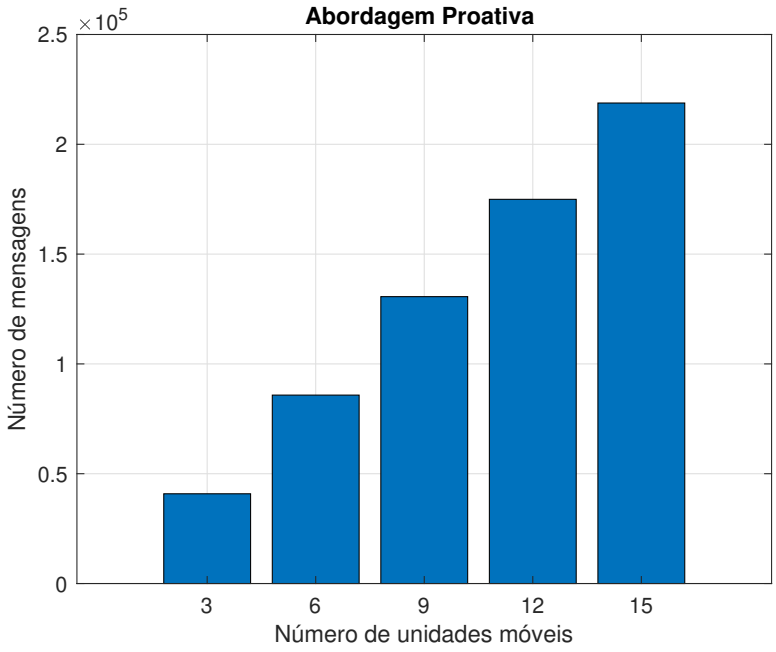


Figura 38 – Número total de mensagens trocadas na abordagem PMS para o serviço policial.

Fonte: Autoria Própria

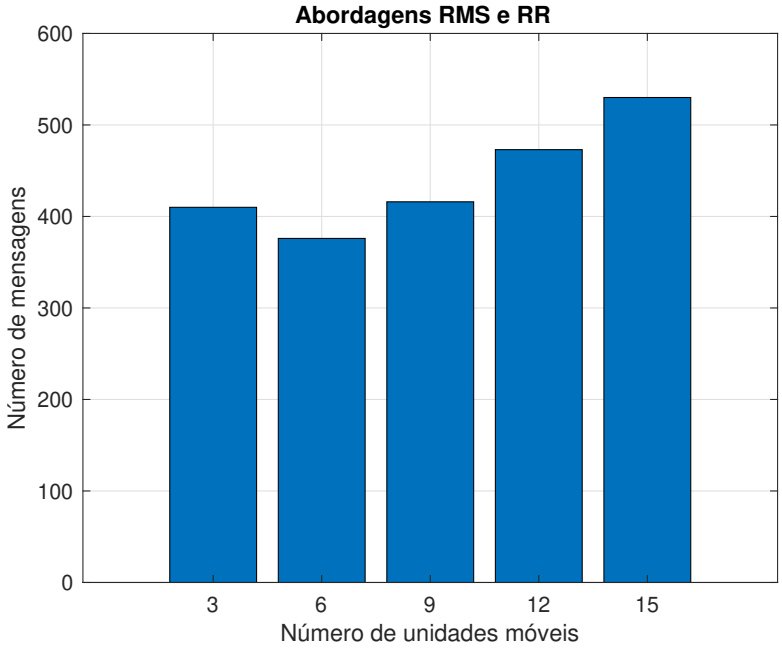


Figura 39 – Média do número total de mensagens trocadas nas abordagens RMS e RR para o serviço policial.

Fonte: Autoria Própria

1. O número de mensagens *subscribe* foi maior na abordagem reativa em relação a abordagem proativa devido ao envio de mensagens *invite-subscribe* pelos *brokers*

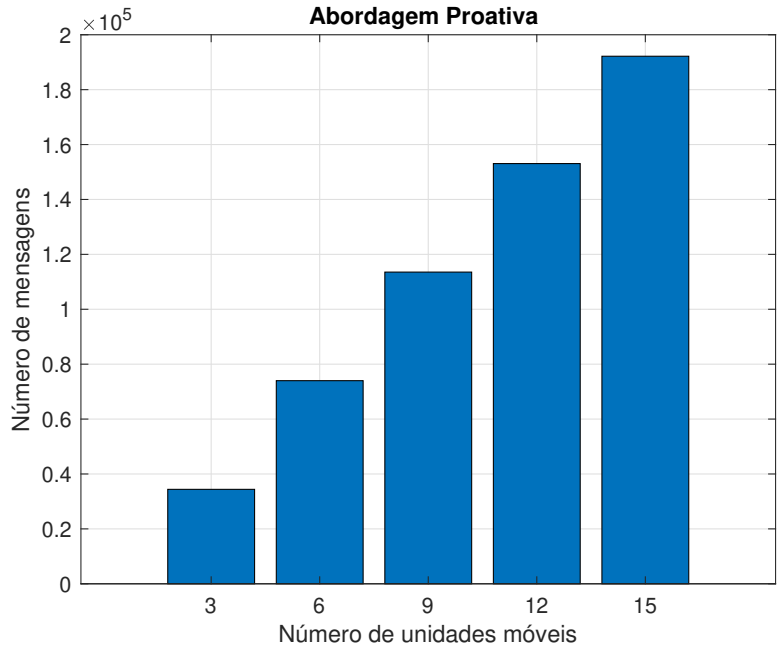


Figura 40 – Número total de mensagens trocadas na abordagem PMS para o serviço de bombeiro.

Fonte: Autoria Própria

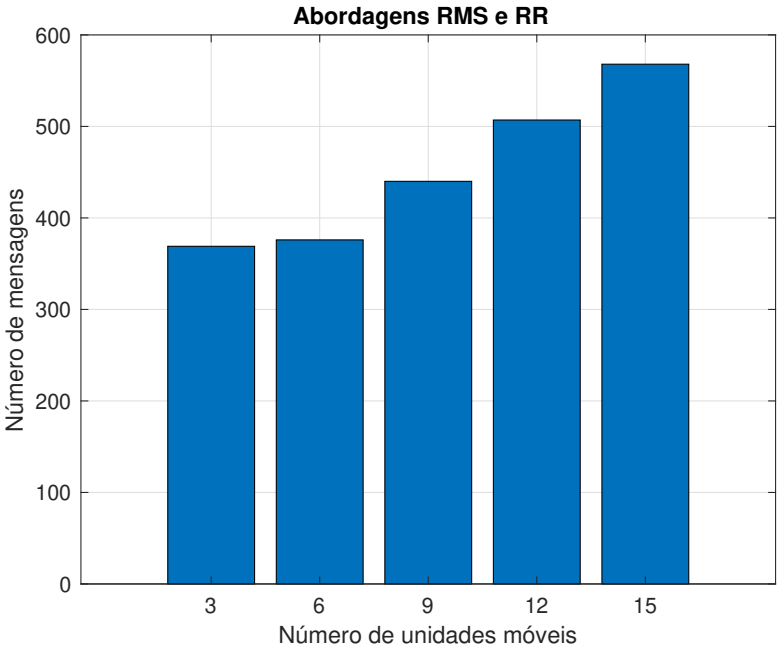


Figura 41 – Média do número total de mensagens trocadas nas abordagens RMS e RR para o serviço de bombeiro.

Fonte: Autoria Própria

publicador e intermediários quando um evento ocorreu.

2. O número de mensagens para atualização de estado (*update-fitness*) foi em média 869 vezes maior na abordagem proativa em relação a abordagem reativa.

3. O número de mensagens *notification-system* foi em média 4 vezes maior na abordagem reativa em relação a abordagem proativa e, como consequência o número de mensagens *ack-notification-system* também.

4. As mensagens *publish*, *ack-publish*, *notification* e *ack-notification* se mantiveram iguais tanto na abordagem proativa quanto na abordagem reativa.

Em todos os serviços simulados, o elevado número de mensagens na abordagem proativa se deve especialmente a questão da atualização de estado que é realizada pelos *brokers*. Toda vez que uma mensagem *subscribe*, *bind*, *unsubscribe* ou *ack-notification* é recebida por um *broker* várias mensagens de *update-fitness* são enviadas entre os *brokers*. Além de que, de minuto em minuto, mensagens de *bind* são enviadas pelos assinantes aos *brokers* das regiões onde se encontram aumentando o tráfego de mensagens entre os assinantes e os *brokers*. O número das mensagens de *bind* poderia ser reduzido se o tempo de envio das mensagens fosse mais espaçado. Contudo, isso comprometeria a eficácia do sistema como um todo porque o estado dos assinantes no momento da ocorrência de um evento poderia estar desatualizado.

Levando-se em consideração o número de mensagens, a abordagem RMS pode ser indicada tanto para um serviço que apresente mobilidade parcial, como no caso dos bombeiros, quanto para um serviço que apresente mobilidade, como é o caso do serviço policial e de transporte. A abordagem PMS é muito mais dispendiosa do que a abordagem RMS em relação ao número de mensagens trocadas na rede por causa da necessidade das atualização de estado dos assinantes em cada *broker*. E isso é custoso para a infraestrutura da rede como um todo. Com o crescimento do número de *brokers* e assinantes um atraso adicional no envio das mensagens na rede poderia acontecer, além do *overhead* de processamento. Logo, a abordagem RMS é a mais indicada se este fator for considerado.

Por fim, o *middleware* EMSS, o qual implementou o modelo de eventos por aptidão de assinante, cumpriu alguns requisitos, dentre eles:

- **Funcionais:**
  - Gerência de eventos.
- **Não funcionais:**

- Escalabilidade (medida em relação ao número de assinantes): o número de assinantes aumentou e o *middleware* EMSS conseguiu fazer a escolha dos melhores assinantes sem comprometer seu funcionamento.
- Confiabilidade (medida por meio da taxa de entrega de eventos considerando a mobilidade dos assinantes): todos os eventos foram entregues aos assinantes, mesmo com os assinantes saindo da região em que se encontravam quando o *broker* precisava os acionar e mesmo quando eventos aconteceram na região do *broker* e não haviam assinantes disponíveis naquela região.

Em relação aos requisitos não funcionais, cabe salientar que como os serviços foram simulados, os dados obtidos e analisados permitiram afirmar que o *middleware* cumpriu os dois requisitos elencados dentros dos parâmetros considerados para cada item.

6 Conclusões

Neste trabalho foi apresentado um modelo de comunicação baseado em eventos por aptidão de assinante (modelo de eventos EMSS), e sua implementação através do *middleware* EMSS. O modelo de eventos foi avaliado por meio de simulação, em uma cidade hipotética, de chamados de emergência para o serviço policial e bombeiro, e chamados para o serviço de transporte. As simulações foram realizadas considerando três abordagens diferentes: **Proativa mais apropriada - PMS**, **Reativa mais apropriada - RMS** e **Reativa aleatória - RR**. Na abordagem PMS, as unidades móveis mais apropriadas foram escolhidas de forma proativa, enquanto que na abordagem RMS as unidades móveis mais aptas foram escolhidas em reação a um chamado de um evento. A abordagem RR difere das outras duas, pois escolhe dentre um conjunto das 50% melhores unidades móveis as necessárias de forma aleatória em resposta a um chamado de evento.

Provou-se que escolher os assinantes mais adequados é melhor do que uma escolha aleatória confirmando, desta forma, a hipótese de pesquisa planejada, a qual conduziu todo o trabalho realizado. Logo, o envio de eventos a um subconjunto de assinantes é melhor do que uma escolha aleatória, dentre um subconjunto dos assinantes mais aptos, pois os eventos serão tratados em um espaço de tempo menor. Com a escolha dos assinantes mais adequados, no caso de uma situação de emergência, o evento é mais rapidamente atendido, salvando vidas. Segundos podem salvar vidas e minutos foram alcançados escolhendo os melhores assinantes. Da mesma forma ocorreu com o serviço de transporte, em que motoristas com um nível de avaliação melhor chegaram mais rapidamente aos chamados, realizando o transporte do passageiro em um tempo menor.

Além disso, a abordagem proativa se mostrou melhor que a reativa, considerando o serviço policial e o serviço de transporte que apresenta mobilidade. Para o bombeiro, a abordagem reativa mostra-se sutilmente melhor que a proativa. Entretanto, salienta-se que as abordagens foram comparadas em um ambiente simulado e talvez, em um ambiente real as abordagens possam se comportar de forma diferente devido as condições reais de trânsito, do sistema viário, do clima, entre outros fatores. Ainda, a abordagem proativa é mais custosa em termos do número de mensagens trocadas entre as entidades do *middleware* EMSS.

Em relação ao *middleware*, o mesmo foi implementado de forma distribuída e a Interface para Programas de Aplicação (*Application Programming Interface* - API) foi desenvolvida dentro do simulador utilizado. A API oferece a implementação do ***broker***, do **assinante** e do **publicador**. Entretanto, como trabalhos futuros esta necessitará de adaptações para ser utilizada em um ambiente real. Por fim, qualquer sistema baseado em

eventos para serviços que necessitem do envio de eventos de forma seletiva pode usar o *middleware* EMSS para ativar os assinantes mais apropriados, como por exemplo:

- Em um serviço de coleta de lixo, o *middleware* seria utilizado para, de acordo com o *fitness* de cada lixeira, traçar a melhor rota de coleta do lixo.
- Em um serviço de estacionamento inteligente, o *middleware* seria usado para, de acordo com o *fitness* de cada vaga de estacionamento, o motorista pudesse escolher aquela mais adequada.
- Em um serviço de monitoração da qualidade do ar, o *middleware* serviria para, de acordo com o *fitness* de cada local da cidade, emitir alertas as autoridades para que, por exemplo, em uma rota com excesso de emissão de CO_2 o fluxo fosse desviado por outro caminho ou então ações preventivas ou educativas incentivando o uso de outros meios de transporte.
- Em um serviço de monitoramento da saúde estrutural de construções, o *middleware* seria utilizado para, de acordo com um limiar de *fitness* de cada construção, as autoridades fossem notificadas sobre a necessidade de intervenção.
- Em um serviço de educação inteligente, o *middleware* seria utilizado para de acordo com o *fitness* de cada sala de aula indicar quais seriam as melhores para receber determinada turma e/ou componente curricular. Ou ainda, de acordo com o *fitness* de cada aluno quais deveriam preencher uma ou mais vagas de estágio.

O *middleware* pode ser expandido para integrar outros sistemas independentes fazendo com que se comuniquem e coexistam dentro de uma Cidade Inteligente. O modelo de eventos EMSS foi um módulo implementado, entretanto outros módulos podem ser adicionados, como por exemplo, módulos de agregação de dados e descoberta de conhecimento. A partir do módulo de descoberta de conhecimento, uma previsão do número de unidades móveis necessárias para uma cidade em função do número de ocorrências poderia ser realizada. Uma modelagem sobre o comportamento dos gráficos, a qual resultaria em outro trabalho, poderia ser realizada visando por exemplo, prever qual o número necessário de unidades móveis para que o TRS ficasse abaixo de certo limiar. Além disso, o modelo de eventos por aptidão de assinante EMSS poderia ser utilizado sobre outro protocolo *publish-subscribe* como, por exemplo, o protocolo MQTT.

Referências

AAZAM, Mohammad; HUH, Eui-Nam. E-hamc: Leveraging fog computing for emergency alert service. In: IEEE. *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. [S.l.], 2015. p. 518–523. Citado 2 vezes nas páginas 33 e 34.

AKRIVOPOULOS, Orestis; CHATZIGIANNAKIS, Ioannis; TSELIOS, Christos; ANTONIOU, Athanasios. On the deployment of healthcare applications over fog computing infrastructure. In: IEEE. *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*. [S.l.], 2017. v. 2, p. 288–293. Citado 2 vezes nas páginas 33 e 34.

AL-FUQAHA, Ala; GUIZANI, Mohsen; MOHAMMADI, Mehdi; ALEDHARI, Mohammed; AYYASH, Moussa. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, IEEE, v. 17, n. 4, p. 2347–2376, 2015. Citado na página 71.

ALKANDARI, A; ALNASHEET, M; ALSHAIKHLI, IFT. Smart cities: survey. *Journal of Advanced Computer Science and Technology Research*, v. 2, n. 2, p. 79–90, 2012. Citado na página 17.

ANTONIĆ, Aleksandar; MARJANOVIĆ, Martina; SKOČIR, Pavle; ŽARKO, Ivana Podnar. Comparison of the cupus middleware and mqtt protocol for smart city services. In: IEEE. *2015 13th International Conference on Telecommunications (ConTEL)*. [S.l.], 2015. p. 1–8. Citado 2 vezes nas páginas 18 e 20.

ARKKO, J; THALER, D; MCPHERSON, D. Ietf rc 7452: architectural considerations in smart object networking. *IETF, Fremont, US*, 2015. Citado 2 vezes nas páginas 29 e 30.

BACON, Jean; MOODY, Ken; BATES, John; MA, Chaoying; MCNEIL, A; SEIDEL, O; SPITERI, M. Generic support for distributed applications. *Computer*, IEEE, v. 33, n. 3, p. 68–76, 2000. Citado na página 56.

BALDONI, Roberto; VIRGILLITO, Antonino. Distributed event routing in publish/subscribe communication systems: a survey. *DIS, Universita di Roma La Sapienza, Tech. Rep*, v. 5, 2005. Citado 8 vezes nas páginas 47, 50, 51, 54, 55, 60, 61 e 63.

BASSI, A; HORN, G. Internet of things in 2020, the european technology platform on smart systems integration (eposs), 2008, 31 p. *Available (referred 13.1. 2017): http://www. smart-systems-integration. org/public/documents/publications/Internet-of-Things_in_2020_ECEPoSS_Workshop_Report_2008_v3. pdf*. Citado na página 27.

BELLAVISTA, Paolo; CORRADI, Antonio; REALE, Andrea. Quality of service in wide scale publish—subscribe systems. *IEEE Communications Surveys & Tutorials*, IEEE, v. 16, n. 3, p. 1591–1616, 2014. Citado na página 56.

BONOMI, Flavio; MILITO, Rodolfo; ZHU, Jiang; ADDEPALLI, Sateesh. Fog computing and its role in the internet of things. In: ACM. *Proceedings of the first edition of the MCC*

workshop on Mobile cloud computing. [S.l.], 2012. p. 13–16. Citado 3 vezes nas páginas 18, 33 e 34.

BOONMA, Pruet; SUZUKI, Junichi. Tinydds: An interoperable and configurable publish/subscribe middleware for wireless sensor networks. In: *Wireless Technologies: Concepts, Methodologies, Tools and Applications*. [S.l.]: IGI Global, 2012. p. 819–846. Citado 4 vezes nas páginas 20, 21, 70 e 73.

CALSAVARA, Alcides; JR, Luiz AP Lima. Routing based on message attraction. In: IEEE. *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*. [S.l.], 2010. p. 189–194. Citado na página 18.

CARZANIGA, Antonio; ROSENBLUM, David S; WOLF, Alexander L. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems (TOCS)*, ACM, v. 19, n. 3, p. 332–383, 2001. Citado 10 vezes nas páginas 20, 21, 47, 50, 51, 57, 58, 59, 60 e 65.

CENEDESE, Angelo; ZANELLA, Andrea; VANGELISTA, Lorenzo; ZORZI, Michele. Padova smart city: An urban internet of things experimentation. In: IEEE. *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. [S.l.], 2014. p. 1–6. Citado 2 vezes nas páginas 17 e 44.

CHAND, Raphaël; FELBER, PA. A scalable protocol for content-based routing in overlay networks. In: IEEE. *Second IEEE International Symposium on Network Computing and Applications, 2003. NCA 2003.* [S.l.], 2003. p. 123–130. Citado 3 vezes nas páginas 61, 62 e 63.

CHENG, Bin; LONGO, Salvatore; CIRILLO, Flavio; BAUER, Martin; KOVACS, Ernoe. Building a big data platform for smart cities: Experience and lessons from santander. In: IEEE. *2015 IEEE International Congress on Big Data*. [S.l.], 2015. p. 592–599. Citado 2 vezes nas páginas 42 e 43.

CHITUMALLA, Pavan Kumar; HARRIS, Douglas; THURAISINGHAM, Bhavani; KHAN, Latifur. Emergency response applications: Dynamic plume modeling and real-time routing. *IEEE Internet Computing*, IEEE, v. 12, n. 1, p. 38–44, 2008. Citado na página 39.

COSTA, Paolo; COULSON, Geoff; GOLD, Richard; LAD, Manish; MASCOLO, Cecilia; MOTTOLA, Luca; PICCO, Gian Pietro; SIVAHARAN, Thirunavukkarasu; WEERASINGHE, Nirmal; ZACHARIADIS, Stefanos. The runes middleware for networked embedded systems and its application in a disaster management scenario. In: IEEE. *Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'07)*. [S.l.], 2007. p. 69–78. Citado 5 vezes nas páginas 20, 21, 59, 69 e 73.

COSTA, Paolo; MIGLIAVACCA, Matteo; PICCO, Gian Pietro; CUGOLA, Gianpaolo. Introducing reliability in content-based publish-subscribe through epidemic algorithms. In: ACM. *Proceedings of the 2nd international workshop on Distributed event-based systems*. [S.l.], 2003. p. 1–8. Citado na página 64.

CRESWELL, John W; CRESWELL, J David. *Research design: Qualitative, quantitative, and mixed methods approaches*. [S.l.]: Sage publications, 2017. Citado na página 100.

CUGOLA, Gianpaolo; JACOBSEN, H et al. Using publish/subscribe middleware for mobile systems. *ACM SIGMOBILE Mobile Computing and Communications Review*, ACM, v. 6, n. 4, p. 25–33, 2002. Citado 3 vezes nas páginas 48, 55 e 56.

DCG, Distributed Computing Group. Sinalgo tutorial. 2015. Disponível em: <https://sinalgo.github.io/tutorial/tuti.html>. Acesso em: 05 de maio de 2015. Citado na página 101.

DIMITRAKOPOULOS, George; DEMESTICHAS, Panagiotis. Intelligent transportation systems. *IEEE Vehicular Technology Magazine*, IEEE, v. 5, n. 1, p. 77–84, 2010. Citado na página 37.

EASTERBROOK, Steve; SINGER, Janice; STOREY, Margaret-Anne; DAMIAN, Daniela. Selecting empirical methods for software engineering research. In: *Guide to advanced empirical software engineering*. [S.l.]: Springer, 2008. p. 285–311. Citado na página 100.

EUGSTER, Patrick Th; FELBER, Pascal A; GUERRAOUI, Rachid; KERMARREC, Anne-Marie. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, ACM, v. 35, n. 2, p. 114–131, 2003. Citado 8 vezes nas páginas 47, 49, 50, 51, 53, 54, 55 e 56.

GUARINO, Nicola. The ontological level. In: _____. *Philosophy and the Cognitive Science*. Vienna: Holder-Pivhler-Tempsky, 1995. p. 443–456. Disponível em: <http://wiki.loa-cnr.it/Papers/OntLev.pdf>. Acesso em: 2 jan. 2012. Citado na página 54.

GUIBERT, Daphné; WU, Jun; HE, Shan; WANG, Meng; LI, Jianhua. Cc-fog: Toward content-centric fog networks for e-health. In: IEEE. *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*. [S.l.], 2017. p. 1–5. Citado na página 34.

GUTIERREZ, Jose M; JENSEN, Michael; HENIUS, Morten; RIAZ, Tahir. Smart waste collection system based on location intelligence. *Procedia Computer Science*, Elsevier, v. 61, p. 120–127, 2015. Citado na página 35.

IBGE. Vamos conhecer o brasil. 2012. Disponível em: <http://7a12.ibge.gov.br/vamos-conhecer-o-brasil/nosso-povo/caracteristicas-da-populacao.html>. Acesso em: 08 de março de 2016. Citado na página 17.

KAFKA, Apache. *Apache kafka a distributed streaming platform*. 2018. Citado 4 vezes nas páginas 20, 21, 71 e 74.

KHAN, Rafiullah; KHAN, Sarmad Ullah; ZAHEER, Rifaqat; KHAN, Shahid. Future internet: the internet of things architecture, possible applications and key challenges. In: IEEE. *2012 10th international conference on frontiers of information technology*. [S.l.], 2012. p. 257–260. Citado 2 vezes nas páginas 31 e 32.

KOPETZ, Hermann. Internet of things. In: *Real-time systems*. [S.l.]: Springer, 2011. p. 307–323. Citado na página 26.

KORTUEM, Gerd; KAWSAR, Fahim; SUNDRAMOORTHY, Vasughi; FITTON, Daniel et al. Smart objects as building blocks for the internet of things. *IEEE Internet Computing*, IEEE Computer Society, v. 14, n. 1, p. 44–51, 2009. Citado na página 26.

LAI, Steven; CAO, Jiannong; ZHENG, Yuan. Psware: A publish/subscribe middleware supporting composite event in wireless sensor network. In: IEEE. *2009 IEEE International Conference on Pervasive Computing and Communications*. [S.l.], 2009. p. 1–6. Citado 5 vezes nas páginas 20, 21, 59, 69 e 73.

LEE, In; LEE, Kyoochun. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, Elsevier, v. 58, n. 4, p. 431–440, 2015. Citado na página 26.

LEONTIADIS, Ilias. Publish/subscribe notification middleware for vehicular networks. In: ACM. *Proceedings of the 4th on Middleware doctoral symposium*. [S.l.], 2007. p. 12. Citado 4 vezes nas páginas 20, 21, 68 e 74.

LOUREIRO, Antonio AF; NOGUEIRA, José Marcos S; RUIZ, Linnyer Beatrys; MINI, Raquel Aparecida de Freitas; NAKAMURA, Eduardo Freire; FIGUEIREDO, Carlos Maurıcio Seródio. Redes de sensores sem fio. In: SN. *Simpósio Brasileiro de Redes de Computadores (SBRC)*. [S.l.], 2003. p. 179–226. Citado na página 26.

MAGNONI, Luca. Modern messaging for distributed sytems. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2015. v. 608, n. 1, p. 012038. Citado 3 vezes nas páginas 49, 53 e 72.

MAJUMDER, Anirban; SHRIVASTAVA, Nisheeth; RASTOGI, Rajeev; SRINIVASAN, Anand. Scalable content-based routing in pub/sub systems. In: IEEE. *IEEE INFOCOM 2009*. [S.l.], 2009. p. 567–575. Citado na página 63.

MARCONI, M de A; LAKATOS, Eva Maria. *Metodologia científica*. [S.l.]: Atlas São Paulo, 2004. v. 4. Citado na página 100.

MEIER, René; CAHILL, Vinny. Steam: Event-based middleware for wireless ad hoc networks. In: IEEE. *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*. [S.l.], 2002. p. 639–644. Citado 7 vezes nas páginas 20, 21, 48, 52, 60, 66 e 73.

MOHANTY, Saraju P; CHOPPALI, Uma; KOUGIANOS, Elias. Everything you wanted to know about smart cities: The internet of things is the backbone. *IEEE Consumer Electronics Magazine*, IEEE, v. 5, n. 3, p. 60–70, 2016. Citado 4 vezes nas páginas 17, 26, 27 e 72.

MOTTOLA, Luca; CUGOLA, Gianpaolo; PICCO, Gian Pietro. A self-repairing tree topology enabling content-based routing in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, IEEE, v. 7, n. 8, p. 946–960, 2008. Citado 2 vezes nas páginas 56 e 64.

MUSOLESI, Mirco; MASCOLO, Cecilia; HAILES, Stephen. EMMA: Epidemic Messaging Middleware for Ad hoc networks. *Personal and Ubiquitous Computing*, v. 10, n. 1, p. 28–36, 2006. ISSN 1617-4917. Disponível em: <http://dx.doi.org/10.1007/s00779-005-0037-4>. Citado 4 vezes nas páginas 20, 21, 67 e 73.

NUAIMI, Eiman Al; NEYADI, Hind Al; MOHAMED, Nader; AL-JAROODI, Jameela. Applications of big data to smart cities. *Journal of Internet Services and Applications*, SpringerOpen, v. 6, n. 1, p. 25, 2015. Citado na página 36.

ONU, Habitat. Estado de las ciudades de américa latina y el caribe 2012. 2012. Disponível em: <http://estaticog1.globo.com/2012/08/21/Estado-de-las-Ciudades-de-America-Latina-y-el-Caribe-2012.pdf>. Acesso em: 05 de março de 2018. Citado na página 17.

OZALP, Nuri; TEKIN, Yasin. Content-based routing for wireless sensor network using intelligent agents. In: IEEE. *2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI)*. [S.l.], 2014. p. 345–350. Citado 3 vezes nas páginas 60, 61 e 63.

PELLICER, Soledad; SANTA, Guadalupe; BLEDA, Andres L; MAESTRE, Rafael; JARA, Antonio J; SKARMETA, Antonio Gomez. A global perspective of smart cities: A survey. In: IEEE. *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. [S.l.], 2013. p. 439–444. Citado na página 41.

PERERA, Charith; QIN, Yongrui; ESTRELLA, Julio C; REIFF-MARGANIEC, Stephan; VASILAKOS, Athanasios V. Fog computing for sustainable smart cities: A survey. *ACM Computing Surveys (CSUR)*, ACM, v. 50, n. 3, p. 32, 2017. Citado na página 72.

PIETZUCH, Peter R; BACON, Jean M. Hermes: A distributed event-based middleware architecture. In: IEEE. *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*. [S.l.], 2002. p. 611–618. Citado 6 vezes nas páginas 20, 21, 55, 65, 73 e 83.

PIVOTAL, RabbitMQ. *Message that just works - RabbitMQ*. 2019. Citado 3 vezes nas páginas 21, 71 e 74.

RAZZAQUE, Mohammad Abdur; MILOJEVIC-JEVRIC, Marija; PALADE, Andrei; CLARKE, Siobhán. Middleware for internet of things: a survey. *IEEE Internet of things journal*, IEEE, v. 3, n. 1, p. 70–95, 2016. Citado 4 vezes nas páginas 27, 31, 47 e 48.

REZENDE, Denis Alcides; MADEIRA, Gilberto dos Santos; MENDES, Leonardo de Souza; BREDA, Gean Davis; ZARPELÃO, Bruno Bogaz; FIGUEIREDO, Frederico de Carvalho. Information and telecommunications project for a digital city: A brazilian case study. *Telematics and Informatics*, Elsevier, v. 31, n. 1, p. 98–114, 2014. Citado na página 46.

RIBEIRO, Admilson RL; SILVA, Fabio; FREITAS, Lilian C; COSTA, João Crisóstomo; FRANCÊS, Carlos R. Sensorbus: a middleware model for wireless sensor networks. In: ACM. *Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking*. [S.l.], 2005. p. 1–9. Citado 4 vezes nas páginas 20, 21, 66 e 73.

RICHARDSON, Alexis et al. Introduction to rabbitmq. *Google UK, available at https://www.rabbitmq.com/resources/google-tech-talk-final/alexis-google-rabbitmq-talk.pdf, retrieved on Mar*, v. 30, p. 33, 2018. Citado na página 20.

ROSE, Karen; ELDRIDGE, Scott; CHAPIN, Lyman. The internet of things: An overview. *The Internet Society (ISOC)*, v. 80, 2015. Citado na página 26.

SANCHEZ, Luis; MUÑOZ, Luis; GALACHE, Jose Antonio; SOTRES, Pablo; SANTANA, Juan R; GUTIERREZ, Veronica; RAMDHANY, Rajiv; GLUHAK, Alex; KRCO, Srdjan; THEODORIDIS, Evangelos et al. Smartsantander: Iot experimentation over a smart

city testbed. *Computer Networks*, Elsevier, v. 61, p. 217–238, 2014. Citado 2 vezes nas páginas 41 e 43.

SANTOS, José; WAUTERS, Tim; VOLCKAERT, Bruno; TURCK, Filip De. Resource provisioning for iot application services in smart cities. In: IEEE. *2017 13th International Conference on Network and Service Management (CNSM)*. [S.l.], 2017. p. 1–9. Citado 2 vezes nas páginas 33 e 34.

SILVA, Bhagya Nathali; KHAN, Murad; HAN, Kijun. Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities. *Sustainable Cities and Society*, Elsevier, v. 38, p. 697–713, 2018. Citado 3 vezes nas páginas 26, 31 e 32.

SILVA, José R; DELICATO, Flávia C; PIRMEZ, Luci; PIRES, Paulo F; PORTOCARRERO, Jesus MT; RODRIGUES, Taniro C; BATISTA, Thais V. PRISMA: A publish-subscribe and resource-oriented middleware for wireless sensor networks. In: *Proceedings of the Tenth Advanced International Conference on Telecommunications, Paris, France*. [S.l.: s.n.], 2014. p. 87–97. Citado 6 vezes nas páginas 20, 21, 59, 70, 73 e 74.

SOUTO, Eduardo; GUIMARÃES, Germano; VASCONCELOS, Glauco; VIEIRA, Mardoqueu; ROSA, Nelson; FERRAZ, Carlos; KELNER, Judith. Mires: a publish/subscribe middleware for sensor networks. *Personal and Ubiquitous Computing*, Springer, v. 10, n. 1, p. 37–44, 2006. ISSN 1617-4917. Disponível em: <http://dx.doi.org/10.1007/s00779-005-0038-3>. Citado 5 vezes nas páginas 20, 21, 60, 68 e 73.

STANDARD, OASIS. Oasis advanced message queuing protocol (amqp) version 1.0. *International Journal of Aerospace Engineering Hindawi www. hindawi. com*, v. 2018, 2012. Citado na página 71.

STANDARD, OASIS. Mqtt version 3.1. 1. *URL http://docs. oasis-open. org/mqtt/mqtt/v3*, v. 1, 2014. Citado na página 72.

SU, Kehua; LI, Jie; FU, Hongbo. Smart city and the applications. In: IEEE. *2011 international conference on electronics, communications and control (ICECC)*. [S.l.], 2011. p. 1028–1031. Citado na página 35.

TSCHOFENIG, Hannes; ARKKO, Jari; THALER, Dave; MCPHERSON, D. Architectural considerations in smart object networking. *RFC 7452*, 2015. Citado 2 vezes nas páginas 29 e 30.

VERMESAN, Ovidiu; FRIESS, Peter; GUILLEMIN, Patrick; GUSMEROLI, Sergio; SUNDMAEKER, Harald; BASSI, Alessandro; JUBERT, Ignacio Soler; MAZURA, Margaretha; HARRISON, Mark; EISENHAUER, Markus et al. Internet of things strategic research roadmap. *Internet of things-global technological and societal trends*, River Publishers, v. 1, n. 2011, p. 9–52, 2011. Citado na página 27.

WANG, Shuo; ZHANG, Xing; ZHANG, Yan; WANG, Lin; YANG, Juwo; WANG, Wenbo. A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access*, IEEE, v. 5, p. 6757–6779, 2017. Citado 2 vezes nas páginas 33 e 34.

YANG, Zhihong; YUE, Yingzhao; YANG, Yu; PENG, Yufeng; WANG, Xiaobo; LIU, Wenji. Study and application on the architecture and key technologies for iot. In: IEEE. *2011 International Conference on Multimedia Technology*. [S.l.], 2011. p. 747–751. Citado na página 32.

YIN, ChuanTao; XIONG, Zhang; CHEN, Hui; WANG, JingYuan; COOPER, Daven; DAVID, Bertrand. A literature survey on smart cities. *Science China Information Sciences*, Springer, v. 58, n. 10, p. 1–18, 2015. Citado na página 26.

ZANELLA, Andrea; BUI, Nicola; CASTELLANI, Angelo; VANGELISTA, Lorenzo; ZORZI, Michele. Internet of things for smart cities. *IEEE Internet of Things journal*, IEEE, v. 1, n. 1, p. 22–32, 2014. Citado 2 vezes nas páginas 17 e 35.

ZELKOWITZ, Marvin V; WALLACE, Dolores. Experimental validation in software engineering. *Information and Software Technology*, Elsevier, v. 39, n. 11, p. 735–743, 1997. Citado na página 100.

ZHAO, Yaxiong; WU, Jie. Building a reliable and high-performance content-based publish/subscribe system. *Journal of Parallel and Distributed Computing*, Elsevier, v. 73, n. 4, p. 371–382, 2013. Citado na página 64.

Anexos

ANEXO A – Dados Coletados

A.1 Corpo de Bombeiros

O Corpo de Bombeiros atende vários tipos de ocorrências, como por exemplo, atendimento pré-hospitalar, acidentes em meio de transporte, combate a incêndio, salvamento, além de prevenção e auxílio. A Corporação que foi solicitado os dados para as simulações do trabalho possui caminhões e ambulâncias ("SIATE"), os quais ficam em três postos fixos de atendimento na cidade. As ambulâncias são utilizadas para o atendimento pré-hospitalar e situações que não requeiram intervenção médica no local da situação de emergência. Ambulâncias de um posto fixo de atendimento mais próximo a ocorrência são sempre acionadas. Entretanto, quando uma situação requer o atendimento médico no local, o Serviço de Atendimento Médico de Urgência (SAMU) é acionado. O SAMU é um serviço do Governo Federal que só é chamado caso a ocorrência necessite de médico. Esse serviço trabalha em conjunto com o governo municipal.

Dados sobre os atendimentos realizados pelo Corpo de Bombeiros podem ser obtidos em www.bombeiros.pr.gov.br. Contudo, essa base não fornece dados sobre o tempo de deslocamento dos veículos da Corporação até a ocorrência, tempo de atendimento e tempo de retorno ao posto de atendimento, além do tempo de deslocamento até o hospital, caso seja necessário. Desta forma, alguns dados foram solicitados ao responsável pelo Corpo de Bombeiros em questão, em outubro de 2017, referente a 2016. Desta forma, apresenta-se gráficos ilustrando o comportamento dos dados obtidos.

A.1.1 Descrição dos dados

A Figura 42 demonstra o percentual de ocorrências por tipo no ano de 2016. Do total de ocorrências, 36% são relacionadas ao atendimento pré-hospitalar (APH), 28% dizem respeito a acidentes em meio de transporte (AMT), 19% são de combate a incêndio (CI), 13% de prevenção e auxílio (PA) e 4% de salvamento (S).

A Tabela 19 mostra o percentual de ocorrências por dia do mês levando em consideração os diversos tipos que o bombeiro atende e os percentuais apresentados na Figura 42. Os dias que apresentaram maior percentual de ocorrências foram os dias 12, 15 e 25.

O gráfico da Figura 43 mostra o percentual de ocorrências por dia da semana. Pode-se perceber que o atendimento pré-hospitalar ocorreu mais as segundas, quartas, sextas e sábados; acidentes em meio de transporte aos sábados, segundas, sextas e domingos; combate a incêndio aos sábados, domingos, segundas e quartas; prevenção e auxílio as

Tabela 19 – Percentual de ocorrências por dia do mês para o serviço bombeiro.

	Percentual de ocorrências				
Dia do mês	**APH**	**AMT**	**CI**	**PA**	**S**
1	4%	3%	3%	3%	4%
2	3%	3%	5%	4%	4%
3	2%	3%	3%	2%	1%
4	3%	3%	3%	3%	2%
5	5%	4%	5%	4%	1%
6	3%	4%	4%	2%	4%
7	4%	3%	3%	3%	2%
8	3%	3%	3%	3%	1%
9	3%	4%	4%	6%	1%
10	3%	4%	3%	3%	5%
11	3%	4%	5%	2%	2%
12	3%	4%	3%	3%	8%
13	4%	2%	3%	3%	2%
14	4%	3%	3%	5%	2%
15	3%	3%	4%	3%	9%
16	3%	4%	4%	3%	3%
17	3%	3%	3%	4%	5%
18	3%	3%	2%	3%	4%
19	4%	3%	3%	3%	4%
20	3%	3%	3%	4%	6%
21	3%	3%	3%	2%	1%
22	3%	3%	2%	4%	2%
23	3%	3%	2%	4%	1%
24	4%	3%	4%	3%	0%
25	3%	3%	4%	3%	14%
26	3%	3%	3%	3%	2%
27	3%	3%	3%	4%	6%
28	4%	4%	4%	3%	1%
29	4%	4%	2%	3%	0%
30	3%	3%	4%	3%	1%
31	2%	2%	2%	1%	2%

Fonte: Autoria Própria

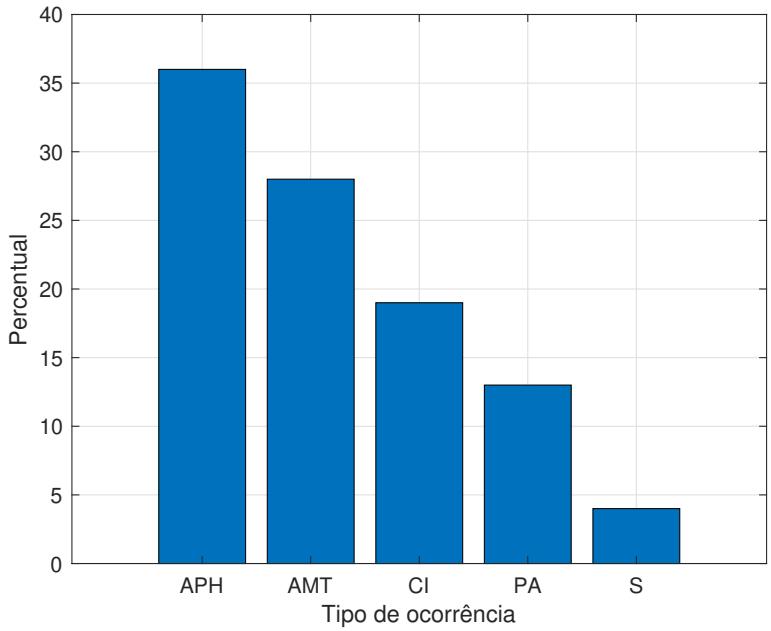


Figura 42 – Percentual de ocorrência por tipo para o serviço bombeiro.

Fonte: Autoria Própria

quintas, terças, quartas e sextas; salvamentos as segundas, sextas, terças e domingos.

A Tabela 20 mostra o percentual de ocorrências por hora do dia. Os horários que mais apresentaram ocorrências foram os horários das 15h e 18h, com uma soma de 40 % entre todas as ocorrências para os dois horários e, em segundo lugar, das 13h e 16h com 37 % das ocorrências para cada um dos horários.

Em relação ao tempo de deslocamento (Figura 44), o tempo médio de deslocamento observado foi de 07 minutos e 06 segundos, como um valor máximo de 47 minutos e um valor mínimo de 01 minuto. A figura mostra a distribuição dos dados de deslocamento. Pode-se perceber que a frequência dos tempos de atendimento diminuem e tendem a zero. Porém, a distribuição é diferente da distribuição normal porque os tempos de atendimento nunca são negativos.

O tempo médio de atendimento foi de 29 minutos e 25 segundos, com um máximo de 8 horas e 30 minutos e com um valor mínimo de 2 minutos e 30 segundos. O gráfico da Figura 45 ilustra a distribuição dos dados de tempo de atendimento (i.e., distribuição beta). Pode-se perceber que a distribuição dos tempos acontece mais próximo da média.

O tempo médio de deslocamento até o hospital foi de 06 minutos e 40 segundos, com um valor máximo de 22 minutos e com um valor mínimo de 02 minutos. O tempo médio de retorno a base foi de 08 minutos e 49 segundos, com um valor máximo de 36 minutos e um valor mínimo de 03 minutos. O número médio de carros médios utilizados por ocorrência foi de 02 carros.

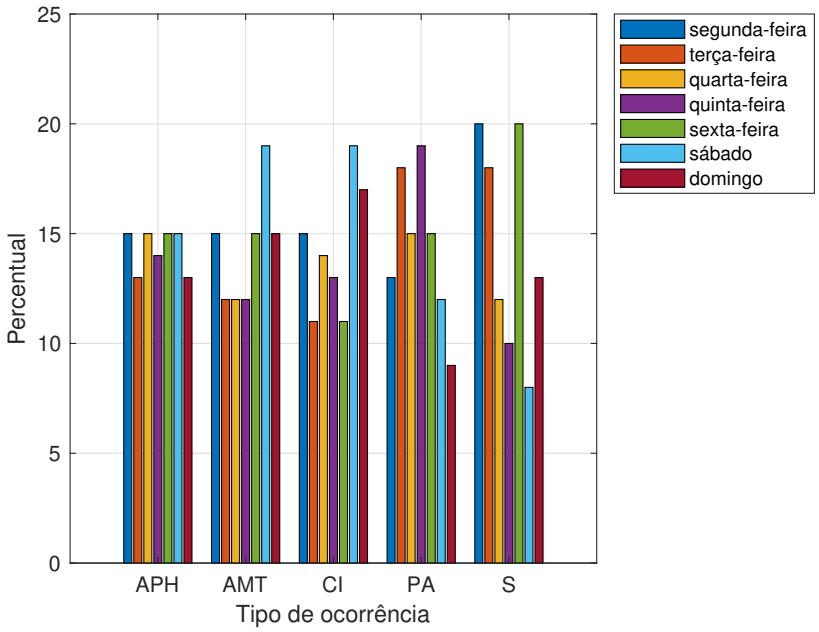


Figura 43 – Percentual de ocorrência por dia da semana para o serviço bombeiro.

Fonte: Autoria Própria

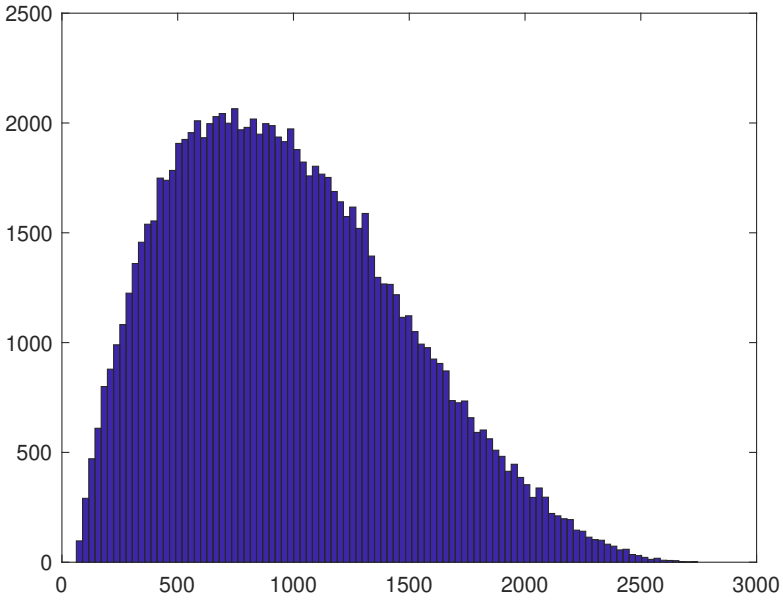


Figura 44 – Distribuição de dados para o tempo de deslocamento do serviço bombeiro.

Fonte: Autoria Própria

Tabela 20 – Percentual de ocorrências por hora do dia para o serviço bombeiro.

	Percentual de ocorrências				
Hora do dia	**APH**	**AMT**	**CI**	**PA**	**S**
0	2%	2%	2%	0%	1%
1	2%	2%	1%	0%	0%
2	1%	1%	1%	0%	0%
3	1%	1%	1%	0%	1%
4	2%	1%	1%	1%	0%
5	1%	1%	1%	0%	1%
6	1%	2%	2%	1%	0%
7	2%	4%	2%	3%	0%
8	4%	5%	1%	10%	4%
9	5%	4%	1%	10%	2%
10	7%	4%	2%	9%	6%
11	6%	6%	4%	4%	2%
12	5%	6%	6%	3%	5%
13	6%	7%	7%	11%	6%
14	6%	5%	9%	7%	9%
15	8%	5%	10%	7%	9%
16	5%	6%	8%	7%	12%
17	6%	8%	6%	6%	11%
18	5%	8%	10%	7%	10%
19	6%	7%	7%	5%	7%
20	6%	3%	6%	3%	8%
21	6%	5%	4%	4%	2%
22	4%	4%	3%	1%	1%
23	3%	2%	4%	1%	3%
24	4%	3%	4%	3%	0%

Fonte: Autoria Própria

A.2 Policial

A polícia atende vários tipos de ocorrências, dentre elas, lesão corporal, furto, roubo e homicídio. Do total de ocorrências, 41% são relacionadas a furtos, 32% relacionadas a lesão corporal, 26% relacionadas a roubos e 1% relacionadas a homicídios. A Figura 46 mostra o percentual de ocorrências policiais por tipo.

A Tabela 21 mostra o percentual dos tipos de ocorrência por mês do ano. Os meses que apresentaram maior percentual de ocorrências foram os meses de outubro, novembro e abril, respectivamente.

Por fim, o tempo mínimo de atendimento de cada ocorrência foi de 20 minutos e o tempo máximo foi de 40 minutos. A Figura 47 mostra a distribuição dos dados em relação

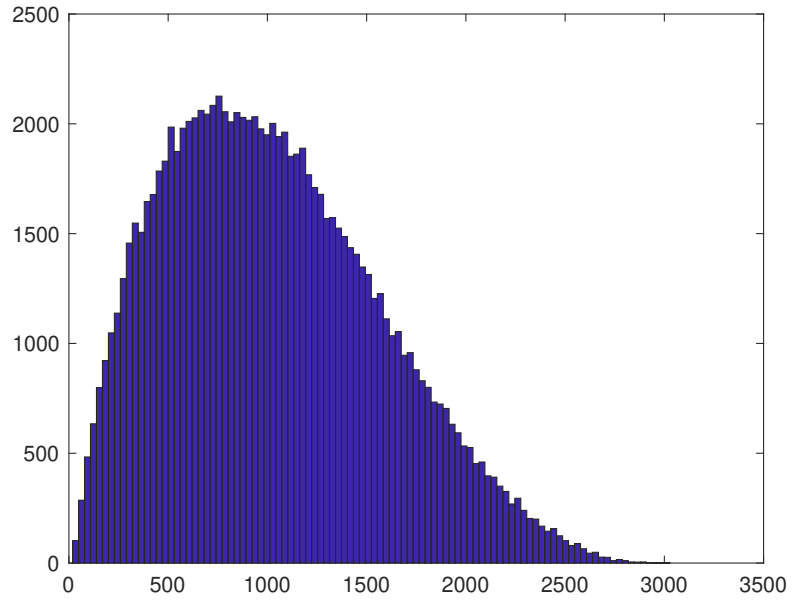


Figura 45 – Distribuição de dados para o tempo de atendimento do serviço bombeiro.

Fonte: Autoria Própria

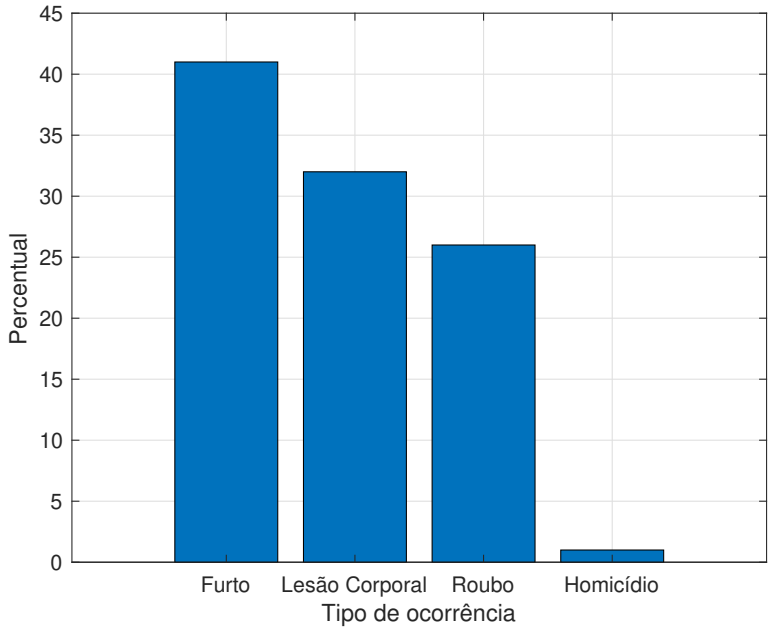


Figura 46 – Percentual de ocorrência por tipo para o serviço policial.

Fonte: Autoria Própria

Tabela 21 – Percentual de ocorrências por mês do ano para o serviço policial.

	Percentual de ocorrências			
Mês	**Furto**	**Lesão Corporal**	**Roubo**	**Homicídio**
1	7%	9%	6%	8%
2	6%	9%	8%	8%
3	6%	10%	6%	6%
4	8%	8%	10%	11%
5	7%	8%	11%	6%
6	10%	7%	9%	6%
7	8%	8%	10 %	8%
8	8%	7%	8%	8%
9	7%	8%	10%	6%
10	10%	9%	7%	17%
11	12%	8%	8%	11%
12	11%	9%	8%	6%

Fonte: Autoria Própria

ao tempo de atendimento das ocorrências policiais.

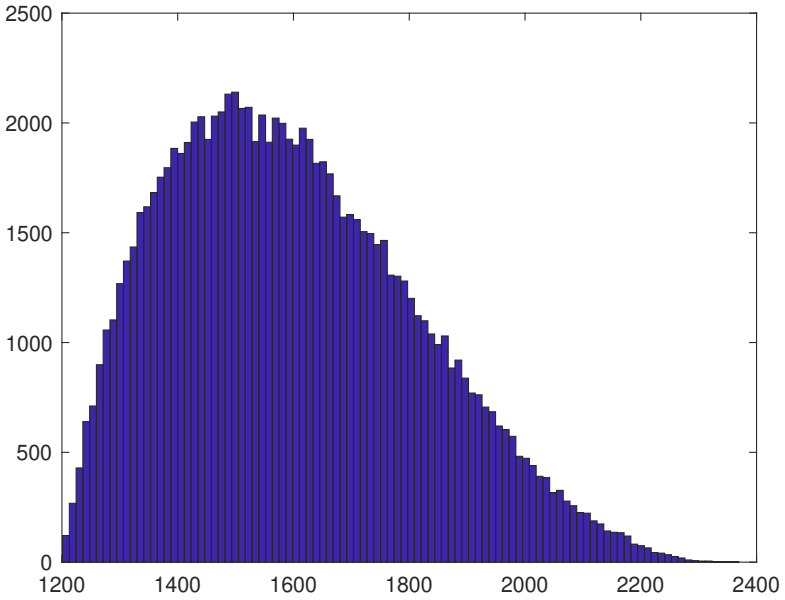


Figura 47 – Distribuição do tempo de atendimento para o serviço policial.

Fonte: Autoria Própria

A.3 Transporte de Passageiros

Para o transporte de passageiros, de acordo com os dados de [1], além de algumas entrevistas com motoristas de Táxi e UBER, chegou-se a conclusão de que em média, em uma cidade de médio porte, um motorista consegue fazer 20 corridas em 12 horas. Tudo depende da duração da corrida, das condições do tempo e de trânsito, do dia da semana, entre outros.

[1] https://ndmais.com.br/noticias/durante-11-horas-motorista-do-uber-faz-ate-25-corridas-por-dia-em-florianopolis/, https://www.campograndenews.com.br/cidades/capital/motoristas-da-uber-dirigem-ate-16h-por-dia-para-ter-lucro-clientes-elogiam e https://uberbra.com/ganhos-diarios-de-um-uber/